DTU Electro
Department of Electrical and Photonics Engineering

Study of the Optimal and Stable Robotic Grasping Using Visual-Tactile Fusion and Machine Learning

Master Thesis

Ramon Fortuny Cuartielles – s212579

Supervisors: Silvia Tolu and Roberto Galeazzi

Study of the Optimal and Stable Robotic Grasping using Visual-Tactile Fusion and Machine Learning

Master Thesis

July 2023

Written by:

Ramon Fortuny Cuartielles

Advisors:

Silvia Tolu, Associate Professor, Ph.D.

Roberto Galeazzi, Associate Professor, Ph.D.

ETCS: 30

Published by: DTU, Department of Electrical Engineering, Elektrovej, Building 326, 2800 Kgs. Lyngby Denmark
www.elektro.dtu.dk

Approval

This thesis has been prepared over five months at the Department of Electrical Engineering and Photonics Engineering, at the Technical University of Denmark, DTU, in partial fulfilment for the degree Master of Science in Electrical Engineering, MSc Eng, with specialisation in Automation and Robot Technology.

It is assumed that the reader has a basic knowledge in the areas of robotics, computer vision and machine learning.

Ramon Fortuny Cuartielles – s212579

……

Signature

27-07-2023

……

Date

Abstract

As the use of robotic manipulators expands across various industries and tasks, the need for efficient and precise object manipulation has become increasingly crucial. The development of a robotic arm capable of manipulating objects with precision relies heavily on accurately identifying the optimal grasping position. Considering this, this project presents a novel method for optimal grasping point calculation and uses machine learning techniques for grasp stability prediction, thus significantly improving the system's performance and efficiency.

The results show evidence of the major improvements achieved thanks to the new developed techniques. First, our system has achieved a success rate of 80% for simple-shaped objects, increasing to a remarkable 92% when inverse kinematics errors were discounted. This success rate outperforms the previous reference of 73% (or 86% without IK errors). Moreover, the system also displayed the same outstanding performance with complex-shaped objects, achieving a success rate of around 90%, whereas the previous benchmark was just around 18% for this type of objects. Apart from the IK-related problems, the only issue that has arisen, preventing us from achieving 100% performance, is a problem with the gripper which, for no apparent reason, does not close to grasp the object when it should.

Overall, these compelling results underscore the efficacy of the innovative methodologies implemented in the system. Despite certain persisting challenges, the research lays the groundwork for future advancements in the field of robotics and automation. The findings of this study have promising applications across a multitude of sectors, particularly in automated manufacturing and logistics, foreseeing a future of higher efficiency and productivity.

Acknowledgements

I am deeply grateful to Silvia Tolu and Roberto Galeazzi, who have acted as my supervisor throughout the development of this project and have provided me with invaluable guidance along the way. Their expertise and support have been fundamental to the success of this thesis.

I would also like to extend special thanks to Nils Meile. His kind introduction to the operation of the robot and the overall system, and guidance during the early stages of this project, were fundamental in helping me to understand the current setup and challenges and work towards improving the system performance.

Finally, I would like to express my most sincere thanks to my family and closest friends. Their unconditional support and motivation have always been a source of strength for me, and their encouragement has been especially meaningful during this journey. I am deeply grateful to them for their belief in my abilities and for their unshakeable faith in my success.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
DoF	Degrees of Freedom
FCI	Franka Control Interface
IK	Inverse Kinematics
KPI	Key Performance Indicator
ML	Machine Learning
PCA	Principal Component Analysis
RDP	Ramer-Douglas-Peucker
ROS	Robot Operating System
SDG	Sustainable Development Goals
UN	United Nations

1. Introduction

1.1. Background

Over the past few years, there has been a notable increase in the use of robotic manipulators in a variety of industries, services, and inspection tasks. This shift is largely driven by the growing trend towards Industry 4.0 [1], which allows manufacturers to integrate new technologies, including robots, Artificial Intelligence (AI), and Machine Learning (ML), that can automate repetitive and monotonous tasks, that for humans can be tedious, and thus increasing operational efficiency and productivity.

Human beings use their senses (such as, vision, hearing, touch, etc.) to interact with and explore their environment. Thus, when we try to perceive an object, we tend to rely on vision to identify it and assess its location. Robots, on the other hand, use sensors to gather information about their surroundings, such as cameras and microphones, among others if needed. In both cases, the ability to accurately sense and perceive the environment is fundamental to performing tasks effectively.

One of the primary challenges of robotic manipulators is to handle objects with precision. This involves finding the exact and best position to grasp an object, which is crucial for successful manipulation. Identifying this optimal location is a challenging but vital aspect that greatly impacts the overall effectiveness and accuracy of the grasping task.

1.2. Problem Statement

As the use of robotic manipulators increases in various industries and tasks, there is a pressing need for accurate and efficient object manipulation. The development of a robotic arm capable of precise object manipulation relies heavily on the accurate identification of the optimal location for grasping an object [2].

In the current landscape, most methodologies for discerning optimal grasping positions fall into two extreme categories: overly simplistic or excessively complex. The simplistic ones tend to base their strategies on direct geometrical properties, such as merely selecting the centre of an object as the grasping point. At the other end of the spectrum, some methods employ intricate neural networks which also have their own drawbacks. The first simpler approach often fails when trying to predict the best grasping location for complex objects due to its lack of sophistication and complexity. In contrast, neural network-based methods face their own challenges such as the requirement of large databases and long training times, limited efficiency in unknown scenarios due to their limited generalisation capability, the need for powerful hardware for training, among others, which makes these solutions not only difficult to implement, but also highly resource-intensive.

Our innovative approach aims to find a balance by developing a fast yet efficient method for determining optimal grasping locations exclusively from vision-based sensor data, in particular, 2D images. This approach is designed to streamline the process, bypassing the pitfalls of

oversimplification and the complexities of neural networks. In addition, our proposed method will incorporate data fusion techniques as well as machine learning algorithms to predict and enhance grip stability. This integrated design will significantly improve the accuracy and speed of operation of the robotic arm, while ensuring a level of consistency and reliability out of the ordinary in current methodologies.

In addition, precise key performance indicators (KPIs) need to be formulated to measure the accuracy and stability of grasping tasks. By addressing these requirements, this method will substantially contribute to improving operational efficiency and productivity in a variety of industrial environments.

1.3. Research Goals and Methods

Based on the problem statement, the aim of this project is to develop a new method to identify the optimal grasping position of objects by vision, as well as to use machine learning techniques to predict and enhance grip stability. To accomplish our objective, the research goal has been divided into several specific sub-objectives. Some of these include, but are not limited to:

- Improve sensor data fusion in terms of speed and quality.
- Develop an accurate and reliable method to identify the optimal grasping position of objects.
- Apply machine learning techniques to improve grasp stability by predicting grasp success.
- Implement, test and evaluate the designed system on an actual robotic arm.
- Determine Key Performance Indicators (KPIs) to assess the accuracy and stability of the grasping task.

1.4. Sustainability

When considering new methods, ideas, and systems, it is crucial to integrate sustainability into the discussion. This inclusion is of paramount importance, not just because of our inherent duty to maintain and enhance the quality and diversity of life on Earth, but also as it aligns with globally recognized goals of sustainable development. The United Nations (UN) has outlined seventeen such goals, designed to guide humanity towards a sustainable future[3].

In the context of this research, these goals serve as a source of inspiration, demonstrating how the utilization of visual-sensor fusion and machine learning techniques in the process of robotic grasping can contribute to sustainable development. While several of these goals can potentially align with the implications of this research, the focus will be placed on Goal 9: Industry, Innovation, and Infrastructure.

The developed method for identifying the optimal grasping position of objects through vision can have significant implications for Industry 4.0. By enhancing the adaptability and versatility of

robotic manipulators, we can reduce the need for producing additional robotic arms, thus fostering sustainable industrialization. This aligns with Goal 9, as it not only promotes resilient infrastructure but also underlines the importance of sustainable industrialization and innovation, all while enhancing productivity and minimizing waste.

Therefore, this research directly supports the UN's ninth sustainable development goal (SDG), underlining the importance of sustainable practices in the ongoing evolution of industry and innovation.

1.5. Thesis Overview

This thesis is structured into six more chapters which are organized as follows. Firstly, Chapter 2 provides a Literature Review, offering a brief overview of current research in the field of robotic grasping and the application of machine learning techniques in this context. Following that, Chapter 3 presents a System Overview, providing a description of the Hardware and Software components utilized in this project. Chapter 4 discusses the Design and Implementation, outlining the approach taken to develop and improve the system's object grasping capabilities. Subsequently, Chapter 5 presents the Results obtained from the various experimental trials conducted in this project. The Discussion in Chapter 6 considers topics unrelated to the experimental results, introduces the Limitations encountered, and outlines potential Future Work directions. Finally, Chapter 7 draws a Conclusion that summarizes the key findings and contributions of the project.

2. Literature Review

In this segment, we will present a comprehensive summary of the pertinent literature that is relevant to the scope of this project. First, we will explain the evolution of robotic grasping from its foundational beginnings to its current complex manifestations. This historical perspective provides us with the necessary context and paves the way for a thorough examination of our main point of reference: the research conducted by Nils Meile [4].

Meile's thesis is used as the starting point for the development of the present project, since the system we will be working with is very similar. However, from his work, potential improvements have been identified and new methods have been developed to improve the performance of the object grasping task, aiming for a higher grasping success rate and a faster and more reliable system.

To conclude, we will introduce decision tree classifiers, a machine learning technique that we will later use to develop a grip stability predictor. Our review will span from its fundamental concept to its operational mechanics, providing a comprehensive understanding of its role in our project.

2.1. Background and Evolution of Robotic Grasping

Robotic grasping has been a focal point of robotics research for many decades due to its fundamental importance in a wide range of applications, from industrial automation to healthcare and domestic environments. The evolution of robotic grasping has seen remarkable progress, driven by advances in robotic technology, computational methods, and the increasing availability of sophisticated sensors.

2.1.1. Early Developments

Early robotic systems designed for object manipulation primarily relied on rigid, pre-programmed routines for specific tasks. These systems, prevalent during the 1960s and 1970s, had limited adaptability to changes in their operating environment. They were mainly used in assembly line environments, where tasks were highly repetitive and predictable. [5]

2.1.2. Advancements in Sensing and Control

As robotic research progressed towards the end of the 20th century, the development of more sophisticated sensors, such as force/torque sensors and vision systems, transformed the capabilities of robotic manipulators. These sensors enabled robots to interact with a wider variety of objects and adapt to changes in their environment in real time.

The introduction of control methodologies like impedance and force control also allowed for safer and more efficient interaction with the environment. Force control played a pivotal role in enabling robots to perform delicate operations like part insertion and assembly. [6]

2.1.3. Vision-Based Robotic Grasping

The advent of vision-based robotic grasping in the late 1990s and early 2000s marked another major milestone in the field. By integrating cameras and advanced image processing algorithms, robots could identify, locate, and grasp a variety of objects. Techniques such as stereo vision and, later, depth sensors, enabled three-dimensional perception of the environment, which significantly enhanced the object manipulation capabilities of robotic systems. [7]

2.1.4. Machine Learning and Robotic Grasping

In the last decade, machine learning, in particular deep learning techniques, have been increasingly applied to robotic grasping. Leveraging large datasets and high computational power, these approaches aim to teach robots to grasp different objects based on learned patterns and features. Reinforcement learning has also been explored as a strategy for robotic grasping, allowing robots to learn successful grasping strategies through trial and error. [8][9]

Despite these advances, robotic grasping remains a challenging problem due to the complexity and variability of the real world. Issues such as occlusions, sensor noise, object variability and real-time processing remain major challenges. However, the field continues to evolve, with more sophisticated sensor fusion techniques, advanced machine learning algorithms and the emergence of soft robotics promising to revolutionise the future of robotic manipulation.

In summary, the evolution of robotic grasping has been characterised by continuous advances in robotic technology, computational methods and sensing capabilities. From the early days of rigid, pre-programmed systems to the intelligent, adaptive manipulators of today, the journey reflects the quest to create machines that can interact with the world with the same dexterity and adaptability as a human hand. [10]

2.2. Using Visual-Tactile Fusion for Intelligent Robotic Grasping

The integration of vision and tactile modalities for improved robotic grasping has been an active field of research, yet it continues to pose challenges that need attention. The research project named “Visual-Tactile Fusion for Intelligent Robotic Grasping” conducted by Nils Meile [4], as part of his master's thesis completed in early 2023, investigated this field in depth. In his project, the primary goal was to develop grasping system that integrates tactile and visual sensors. This fusion was expected to provide a richer data set, since information from tactile sensors would complement that information provided by a purely visual system, in the hope of improving object manipulation.

2.2.1. Systems Integration

In his project, Meile first had to take the crucial step of integrating several systems from different research projects into a cohesive unit. This involved both hardware and software.

On the hardware side, the integration consisted of attaching a robotic gripper supplied by Graspian Aps to a Franka Emika Panda 7 degrees of freedom (DoF) robotic arm. This was a fundamental step, as it allowed the subsequent study of the concept of tactile and visual sensor fusion explored in his research.

In terms of software integration, Meile also carried on from the development of other master's thesis by previous students in this field. A key example was "A Model-Free Predictive Control Framework for Trajectory Optimisation with a Collaborative Robot"[11] by Christian Kampp Kruuse in 2022, which developed the full control system of the robot. Last, the gripper's own software was also integrated into the overall system. All these software systems were then brought together harmoniously in the Robot Operating System (ROS) framework.

The consolidation of such diverse systems into a single functional unit was crucial for Meile's research, as it paved the way for the development and actual testing of the gripper controller based on the fusion of tactile and visual sensors.

2.2.2. Inverse Kinematics Failures

Partly related to the topic of the previous section, in the execution of this project, we anticipate encountering some challenges like those faced by Meile, particularly in the realm of Inverse Kinematics (IK). Meile's system's integration of the control system designed by student Christian Kampp Kruuse, also incorporated a TRAC-IK solver. Although this solver was functional, it was not without some flaws.

In Meile's experiments, issues were observed with the IK-solver, including erratic shaking of the robotic arm when attempting movements towards specific areas, occasional overshooting of joint limits, and complete IK-solver failures that resulted in the shutdown of the ROS-node of the controller. These failures tended to occur when the robotic arm was operating near its extreme range of motion, such as when attempting to reach objects in the far front-left side of the operating area, which is the rightmost part of the conveyor belt.

In the context of our project, it is important to clarify that solving these IK issues is not our primary objective. We do acknowledge that these complications can interfere with successful object grasping. However, we have been able to partially mitigate these effects indirectly through the application of machine learning techniques.

As we will detail further in section 5.2.2, the Decision Tree Classifier that we will generate will help us discern certain attributes that contribute to unsuccessful grasping attempts, aside from the inherent IK failures. Thus, by recognizing and avoiding these unfavourable conditions, we will be able to decrease the rate of grasp failures, even in areas traditionally prone to IK failures.

While we cannot eliminate the occurrence of IK failures, our project has been able to leverage Machine Learning to mitigate some of these problems, thereby also improving the overall grasping success rate.

2.2.3. Identification of the Grasping Point for Objects

In his endeavour to identify in real time the best grasping location of objects, Meile already identified some limitations in the mono-static camera system we will use, since it lacks both colour and depth perception. Furthermore, the camera setup only gives us a top view of the conveyor belt and the objects to be grasped, which also limited the methods available for object detection.

Meile also realized that the requirement for real-time visual processing introduced substantial complications, as object detection algorithms need to process images around the live video frame rate, which imposes computational limits [12]. Therefore, it's generally not feasible to identify all objects on an image in real-time. Research findings have indicated that it's considerably more feasible to pinpoint specific features of an object, such as its edges, symmetry, corners among others instead [13]. In Meile's case, the object shapes, whether circular or rectangular, and the orientation of the rectangular shapes, were the key features to be extracted.

For distinguishing between circular and rectangular objects, Meile used the Ramer-Douglas-Peucker (RDP) algorithm. The RDP algorithm reduces a detailed curve into a simpler series of points by creating lines between points and removing those points that do not significantly affect the shape of the curve [14]. Using this algorithm, Meile took advantage of the fact that rectangles can be described with fewer data points than a circle to distinguish between them. Thus, the number of data points required by the RDP algorithm became a distinguishing factor.

Then, for rectangular objects and to determine their orientation, Meile utilized Principal Component Analysis (PCA). PCA reduces the dimensions of a multivariate dataset by defining a new reduced set of uncorrelated variables called principal components [15]. These components can provide a simplified representation of the original dataset. In this case, this was used to extract the general direction or orientation of rectangular objects.

The goal of PCA is to capture as much of the data's variance as possible with fewer dimensions. It achieves this by establishing a new coordinate system, where the basis vectors, or principal components, are oriented in the directions of maximum variance. In a 2D dataset, two orthogonal principal components are identified. The first component, corresponding to the larger eigenvalue of the covariance matrix, aligns with the greatest variance in the dataset. The second component, orthogonal to the first, captures the next most significant variance.

A visual representation of the way that PCA reduces the dimensions of a given dataset can be seen in the following image:

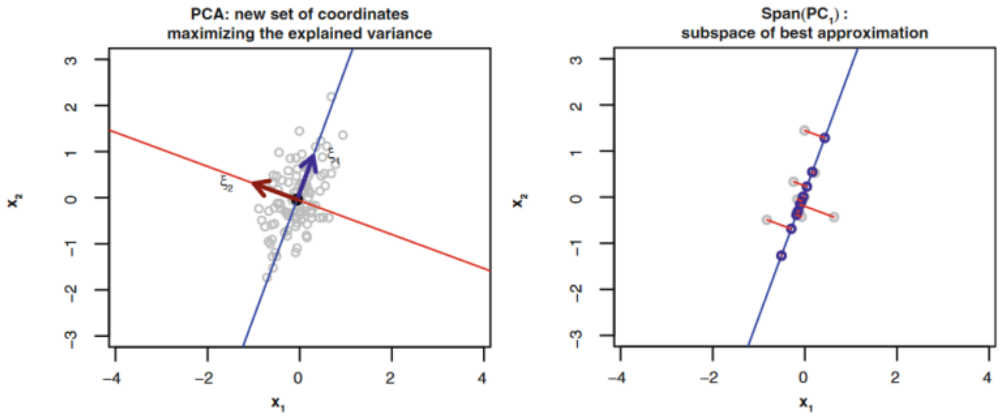

Figure 1. Dimensionality reduction through principal component analysis (PCA)

In terms of determining the optimal grasping point, regardless of whether the objects were rectangular or circular, Meile deployed a method centred around the concept of the object's centre of mass. This approach was adopted for both object shapes, considering the centre of mass as a simple but good point for grasping the objects.

2.2.4. Evaluation of Meile's Method for Identifying the Grasping Point of Objects and Potential Enhancements

The methods employed by Meile provided an efficient way of identifying the object shapes, their locations, and their orientation, in case of rectangular objects. However, this oversimplistic approach was not good enough for complex objects that were neither circular nor rectangular, as the centre of mass was not actually the correct grasping location. In fact, this was clearly observed when trying to grasp a slightly more unusual object, such as a banana, in the testing and results phase of his thesis. In the specific case of the banana, it could be observed that the success rate in grasping this object was only 18%, compared to an average of 86% in the other experiments.

Therefore, my research will build on the foundations laid by Meile's work, where, as has been seen, there is potential to improve the process of identifying the best grasping location of the object, which forms the central focus of my project, thus achieving a better success rate of the grasping task. On the other hand, my project will also seek to improve data fusion from different sensors (by reducing some bottlenecks), as well as making the overall system more robust (less prone to errors and failures).

By addressing these areas, the goal is to advance the tactile-visual sensor fusion approach to robotic grasping, enabling more effective and efficient object manipulation.

2.3. Decision Tree Classifiers

In this section, we introduce Decision Tree Classifiers, a machine learning technique employed in our work for predicting the success or failure of grasping tasks. We will delve into the mechanism of how they operate and explain why they are an effective choice for our specific application. This understanding serves as the foundation for their later implementation in Section 4.4, Grasp

Stability Prediction using Machine Learning Techniques, with the goal of improving the performance of the robotic system.

2.3.1. Introduction to Decision Tree Classifier

A Decision Tree Classifier is a form of supervised machine learning model that represents decisions and their possible results visually and analytically. It's essentially a flowchart-like structure in which each internal node represents a test on an attribute (e.g., “Grasping Rotation”), each branch signifies an outcome of the test, and each leaf node (terminal node) holds a class label (i.e., “Success” or “Failure”).

For a better understanding of Decision Tree Classifiers, an illustrative representation is provided below. This diagram highlights the essential components of a Decision Tree: the root node, the decision nodes and the leaf nodes. In this model, each decision and each root node represents an attribute of the dataset under evaluation, splitting the data according to the satisfaction of the attribute condition in two branches, yes or no. Finally, the leaf nodes represent the final classified classes, formed by the combination of the individual evaluations up to that node.

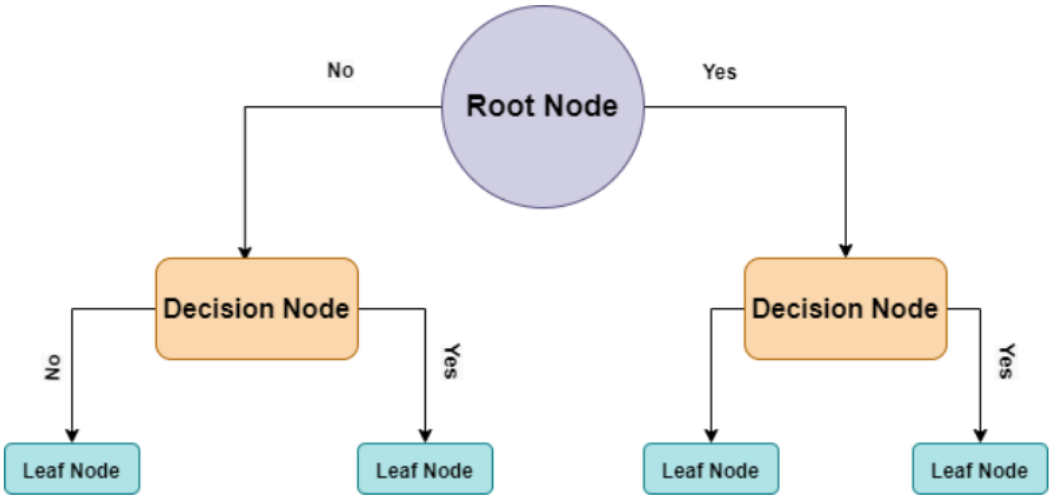

Figure 2. Visualisation of the structure of the decision tree classifier

The key reason for choosing a Decision Tree Classifier for this project is its simplicity and interpretability. Decision Trees are easy to understand and visualize, which makes them particularly valuable for our use case, where the aim is to predict stable object grasping. Moreover, they are capable of handling both numerical and categorical data, although in our case the inputs are only numerical data. Furthermore, decision trees implicitly perform feature selection, giving importance to the most significant variables first. This will become beneficial as it will help us understand what features contribute the most to a successful grasp.

2.3.2. Understanding Gini Impurity

A key aspect of the operation of a Decision Tree Classifier is the proper selection of an attribute at each node to split the data. If no perfect attribute can lead to pure leaf nodes (i.e., leaf nodes

representing only one class, in our case, either 100% of Success or 100% of Failure), an impurity measure guides the choice of the "best" attribute to split the data into the purest possible nodes. [16]

In this case, Gini impurity will be our selected impurity measure. Essentially, Gini impurity quantifies the purity of the split in a Decision Tree Classifier. It will help us to identify the most efficient attribute splitter, enabling the construction of a decision tree that maximises class purity at each split.

Mathematically, the Gini impurity for a leaf is computed as:

$$\boldsymbol{Gini_{leaf}} = 1 - \sum_{i=1}^{j} (p_i)^2 \tag{1}$$

Where j is the total number of classes (in our case 2, Success or Failure), and p_i the probability of each of the corresponding classes.

But that only was the Gini impurity of a single leaf for a given node that evaluates the separation of the data according to certain condition applied to one attribute (input) of the dataset. For obtaining the Gini impurity of the split of a node (given a certain condition to an attribute) into 2 leaves, we just need to compute the weighted average of the Gini impurities for the leaves, as shown in the formula below:

$$\boldsymbol{Gini_{node}} = Gini_{leaf\ 1} \cdot \frac{N.points_{leaf\ 1}}{Total\ N.points_{node}} + Gini_{leaf\ 2} \cdot \frac{N.points_{leaf\ 2}}{Total\ N.points_{node}} \tag{2}$$

Where the $Total\ N.points_{node}$ is basically the $N.points_{leaf\ 1} + N.points_{leaf\ 2}$.

In order to provide a much better understanding, a very simple dataset table is shown below as an example, where we can see 3 different attributes (A, B, C) and a resulting class (Result) which will be either Success or Failure:

Likes A?	Likes B?	Number of C	Result
Yes	No	1	Success
Yes	Yes	3	Success
No	No	5	Failure
Yes	No	7	Failure
Yes	No	9	Success
Yes	Yes	13	Success

Table 1. Example of dataset to illustrate Gini impurity calculation

Let's now calculate the Gini impurity of the possible root nodes. Let's pick, for instance, the first attribute, "Likes A?", to evaluate how well we can split the data. As it can be seen, this attribute will have 2 possible values of Yes or No.

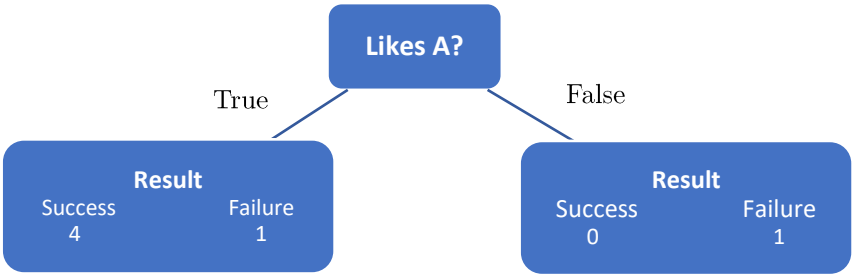

Figure 3. Example of decision tree split based on an attribute

So, based on the previous split, first we will calculate the Gini impurity of each resulting node. Thus, we will have:

$$\boldsymbol{Gini_{leaf\ True}} = 1 - \sum_{i=1}^{j} (p_{Result})^2 = 1 - (p_{Success})^2 - (p_{Failure})^2 = 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2 = 0{,}32 \tag{3}$$

$$\boldsymbol{Gini_{leaf\ False}} = 1 - \sum_{i=1}^{j} (p_{Result})^2 = 1 - (p_{Success})^2 - (p_{Failure})^2 = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2 = 0 \tag{4}$$

$$\boldsymbol{Gini_{Likes\ A?}} = Gini_{leaf\ True} \cdot \frac{N.points_{leaf\ True}}{Total\ N.points_{node}} + Gini_{leaf\ False} \cdot \frac{N.points_{leaf\ False}}{Total\ N.points_{node}} = \\ = 0{,}32 \cdot \frac{5}{6} + 0 \cdot \frac{1}{6} = 0{,}27 \tag{5}$$

Now that we have obtained the Gini impurity for the first attribute, we would find the Gini impurity for the other ones and compare it between each other. The Gini impurity always lies between 0 and 0,5, with 0 being the best-case scenario where all elements at a node belong to a single class, resulting in pure class segregation (like in leaf False for the first attribute, as seen).

Conversely, a Gini impurity of 0.5 represents the worst-case scenario, indicating a 50-50 class split at a node and therefore, the highest uncertainty. Thus, at the end we will select the split which has the lowest Gini impurity, which will give us the best (the purest) split of data in order to at the end classify better the Result class (Success or Failure) based on the splits of data done in the different attributes.

In the exact same way, if we compute the Gini impurity for attribute "Likes B?", we obtain a Gini impurity of 0,33, which is worse than the Gini impurity for the first attribute. If these 2 were the only 2 attributes to evaluate, we would pick the first one to split the data since it returns us a purer split. However, the third attribute, "Number of C", has not been considered yet.

Third attribute, as it can be seen, doesn't have 2 possible values as the previous ones had. Instead, this one consists of numerical values. In this situation, what we will do to calculate the Gini impurity, is to calculate the Gini impurity of each of the average values between the different possible values taken by the attribute. So, if we look again at Table 1, we will first obtain the mean values of the different values of the attribute as follows:

Number of C	Result
1	Success
2	
3	Success
4	
5	Failure
2	
7	Failure
8	
9	Success
11	
13	Success

Figure 4. Average values between the different possible values of numerical attributes

Now, we can calculate the Gini impurity for each of the average values. Just as before, we will show a representation of the division of the nodes simply for the first case as an example:

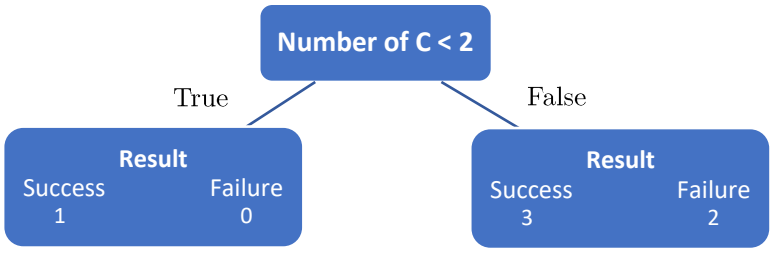

Figure 5. Example of decision tree split based on a numerical attribute

$$\boldsymbol{Gini_{N.of\ C<2}} = Gini_{True} \cdot \frac{N.points_{True}}{Total\ N.points} + Gini_{False} \cdot \frac{N.points_{False}}{Total\ N.points} = \\ = 0 \cdot \frac{1}{6} + \left[1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right] \cdot \frac{5}{6} = 0{,}4 \tag{6}$$

As we have done for the mean value of 2 of the "C number", we will then calculate the remaining Gini impurities for the remaining values:

$$\boldsymbol{Gini_{N.of\ C<4}} = 0 \cdot \frac{2}{6} + 0{,}5 \cdot \frac{4}{6} = 0{,}33 \tag{7}$$

$$\boldsymbol{Gini_{N.of\ C<6}} = 0{,}44 \cdot \frac{3}{6} + 0{,}44 \cdot \frac{3}{6} = 0{,}44 \tag{8}$$

$$\boldsymbol{Gini_{N.of\ C<8}} = 0{,}5 \cdot \frac{4}{6} + 0 \cdot \frac{2}{6} = 0{,}33 \tag{9}$$

$$\boldsymbol{Gini_{N.of\ C<11}} = 0{,}48 \cdot \frac{5}{6} + 0 \cdot \frac{1}{6} = 0{,}4 \tag{10}$$

Now that all these values have been calculated, they can be compared together with the previously obtained value of the Gini impurity of attribute A (0,27) as well as with the Gini impurity of attribute B (0,33), picking thus the lowest of these values, that is, 0,27 (attribute A).

If we recall, Figure 3 already showed how the nodes would have been split in the end. In that image we can see that the resulting node on the right was a pure node, since only one class (Failure) jumped into this split according to the criteria applied to attribute A. However, the left-hand node was not completely pure, leaving room to split again into purer splits. Thus, if we wanted to continue the separation of the data points for better categorisation, we would then evaluate the other attributes not yet evaluated at this depth of the decision tree (i.e., attribute B and C), this time only for the set of points belonging to the new node.

This process would be repeated either until completely pure nodes are reached or until a certain level of desired depth in the decision tree is reached.

3. System Overview

This system overview section provides a comprehensive insight into the principal hardware and software components used in the project. First, we will review the main hardware components, consisting of a 7-DoF Franka Emika Panda robot arm, a BlueFox3 Camera, and a Graspian tactile gripper, each of which makes a unique contribution to the project. Additionally, we utilize several distinct objects for experimental manipulation, representing various real-world challenges. On the software side, the project leverages the powerful Robotic Operating System (ROS) framework, using Python and C++ programming languages, to ensure efficient programming, scalability, and design flexibility. Altogether, the project combines these elements to create a versatile and precise robotic system, capable of performing intricate manipulation tasks.

3.1. Hardware

3.1.1. Franka Emika Panda Robot Arm

In this project, one of the essential components employed is the Franka Emika Panda robotic arm, a 7-degrees-of-freedom (7-DoF) mechanism capable of handling a payload of up to 3 kg and extending its reach to 855 mm. Known for its interactive functions, the robot features an intuitive graphical user interface (GUI) that makes it even easier to operate, allowing the user to easily interact with the robot and program, for example to design simple task flows. [17]

Figure 6. Franka Emika Panda robot arm

However, it is important to note that despite the ease of use of the GUI, our project will work on the 1 kHz Franka Control Interface (FCI), an additional feature of the Franka Emika robot. The FCI is an optimal platform for exploring and testing various elements, such as control and motion algorithms, grasping tactics, interaction scenarios and machine learning as this interface enables a fast, low-level bidirectional connection with the robot's arm and hand, extending its versatility and responsiveness for intricate tasks. In addition, the manufacturer also offers a pair of practical software libraries, "franka_ros" and "libfranka", which significantly simplify the establishment of a functional connection to the robot via ROS (Robotic Operating System), allowing easy access to critical data such as joint states and end-effector position in world coordinates.

The robot also includes several other functionalities and features, such as the inclusion of a "guiding mode" in which by pressing a couple of buttons on the robot you can operate the robot freely and without resistance, as well as an emergency stop button to stop the robot at any moment.

It should also be noted that although the Franka Emika robot has a native robotic gripper, the project did not use it because, as already mentioned, our project is based on the previous work done by Meile [4]. In his case, the decision to use another robotic gripper instead was because the one provided by Franka Emika lacked real-time control support and used another different interface, which would complicate working with it.

3.1.2. Graspian Tactile Gripper

Figure 7. Graspian tactile

In this project, the robotic gripper employed is manufactured by the company Graspian, as it has tactile sensing capabilities. This, as shown in Figure 7, is a non-industrial 1-DoF robotic gripper, consisting of two fingers, each with a resistive tactile sensor consisting of seven piezoresistive taxels (TActile piXEL) covering an area of 4,1 cm by 3,05 cm. These taxels are enveloped in a foamy substance that serves as a protective and damping layer to compensate for the uneven force distribution.

When the finger experiences pressure, there is a change in the resistivity of the taxel, which causes a decrease in voltage. This voltage fluctuation can be used to calculate the force exerted based on a conductivity/force model. Communication between the clamp and a computer is established via a USB connection to a Teensy board, using the RS-232 serial protocol. Accompanying the touch sensor is a pre-developed embedded software, built on PlatformIO, used throughout this project. The software allows the transmission of various commands to the gripper, initiating specific actions and sharing sensor information back with the computer.

3.1.3. BlueFox3 Camera

The project employs a stationary Matrix Vision's BlueFox3 camera similar to the one shown in Figure 8. As an industrial camera, it's designed to consistently provide high-quality imaging for diverse applications, including continuous 24/7 operations. The camara will provide us a top view of the conveyor belt where objects will be placed, capturing monochrome images, which, while they offer high levels of detail, lack colour representation and depth information, complicating the task of obtaining the best grasping location for the objects.

Figure 8. Matrix Vision's BlueFOX3 camera

Calibration of the camera wasn't a requirement for this project, as this process had already been undertaken by Christian Kampp Kruuse [11]. Likewise, the integration of the camera into the ROS environment

with the relevant installation of drivers and packages necessary for imaging was carried out by Niels Meile.

3.1.4. Objects

To run the different experiments, different types of objects have been chosen. Among the many different possible objects, it has been decided to keep the 5 main objects already used by Meile in his project in order to be able to compare the results of the various improvements made to the system. These 5 objects are a sponge, a rectangular wood block, a Christmas ornament, a tennis ball and a soft ball as shown below.

Figure 9. Selected objects for comparative experiment results

However, new objects of a more complex shape, other than circular or rectangular objects, were also added to evaluate and demonstrate the capabilities of the newly developed method for obtaining the grasping location of objects. In fact, we will use a banana among other ones, which Meile already tried to grasp in his project [4], but then realised that his method of obtaining the grasping position was not suitable for objects with more complicated shapes. Some of the other objects used in this part are pieces of polystyrene cut in such a way as to create more complex shapes such as U-shaped or O-shaped objects, as seen in the following picture.

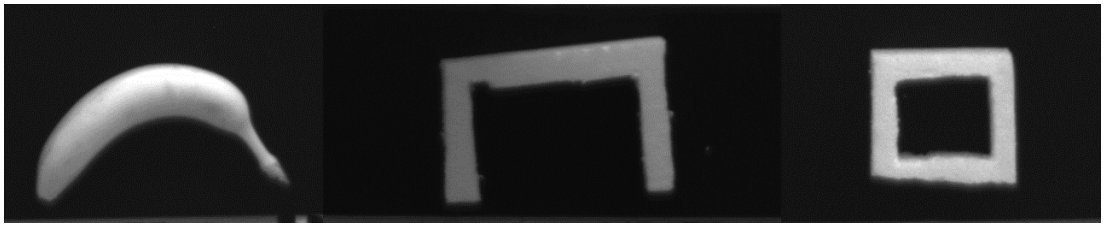

Figure 10. Selected objects of complex shape

The only limitation that remains in both projects is when choosing the height of the objects, since, as mentioned above, a single camera without sensors capable of estimating the depth makes it impossible to adapt the height of the gripper at the end of the robot arm that will grasp the object in question. For this reason, objects cannot be too high, otherwise they will collide with the gripper.

3.2. Software

3.2.1. Robotic Operating System

In this project, the Robot Operating System (ROS) serves as the core framework for managing the complex behaviour of the robot. ROS is not a conventional operating system, but rather an open-source collection of tools, libraries, and protocols designed to streamline the process of creating intricate and reliable robotic applications across a multitude of platforms. [18]

ROS is open-source and is used to manage various aspects of the robot's functionality. It's best described as a middleware because it helps to manage communication and control between the various software applications - running on the different hardware mentioned - that make up the operation of the robot. As such, ROS plays an important role in operating and controlling the various components used in this project in one place.

In ROS, the main components are Nodes and Topics. Nodes are basic units of computation managed by ROS, which enable communication by publishing or subscribing to Topics. Topics act as channels where Nodes publish and receive messages, facilitating communication and information exchange between them. The Master plays a crucial role, offering name registration and lookup services that enable Nodes to identify each other and establish communication.

3.2.2. Programming Languages

ROS provides a flexible programming interface that primarily supports C++ and Python. C++ is typically used for tasks requiring high computational performance, such as real-time signal processing, due to its efficient handling of low-level computations and direct access to underlying data structures. Python, with its simplicity and readability, is commonly used for high-level software, testing, and user interactions where rapid prototyping and scripting are required. Additionally, other languages like Lisp or Java are supported, although their use within the ROS community is less common. In our project, we effectively leveraged both C++ and Python, making use of their respective strengths for specific tasks.

4. Design and Implementation

In this chapter, we take an in-depth look at all the different processes and methodologies employed to convert raw camera images into the execution of the specific robot commands needed to successfully grasp and manipulate the objects.

This process comprises several stages, each unique in its contribution to the final product. The path to successful object manipulation begins with images preprocessing enhancing raw camera feed for effective object detection in the subsequent step. Following object detection, a newly devised method is employed to determine the most optimal grasping point for each detected object. Leveraging machine learning techniques, we then predict the success rate of the grasping task, assessing its stability. Concluding the process, the necessary commands are sent to the robot to execute the task of object manipulation, completing the entire sequence of operations needed to move the object.

Figure 11 below provides a comprehensive overview of this entire sequence of operations, covering every step from image preprocessing to final object manipulation.

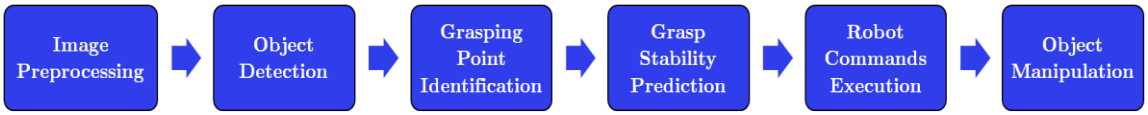

Figure 11. Process flow diagram of the implemented design

4.1. Preprocessing Techniques for Improved Image Analysis

The first crucial step in the object manipulation process is the image preprocessing stage. The purpose of this step is to refine the raw images received from the camera, reducing noise and irrelevant details, which enables more precise object detection and differentiation in the subsequent stages. The entire procedure is broken down into several key steps, each employing different image processing techniques. For this purpose, we made significant use of OpenCV functions, a powerful open-source computer vision and machine learning software library.

The raw images we receive from the camera are initially quite cluttered with unnecessary details and noise, as can be seen in Figure 12. To effectively focus our object detection and differentiation efforts, we perform a series of refinements on these images.

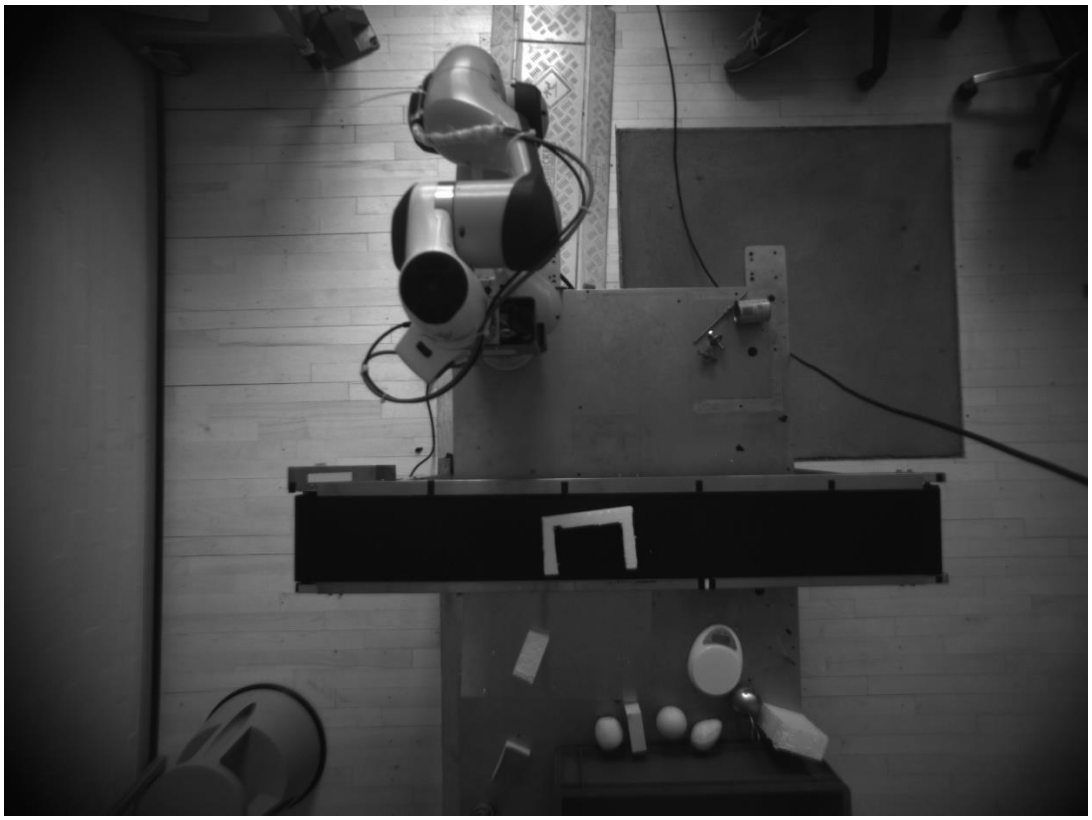

Figure 12. Raw image received from the camera

Firstly, to restrict our focus to the region of interest, the conveyor belt in this case, we mask all other sections of the image by overlaying black rectangles over them. This process, depicted in Figure 13, helps us eliminate distractions from the image, thereby optimizing the efficiency of the object detection process that follows.

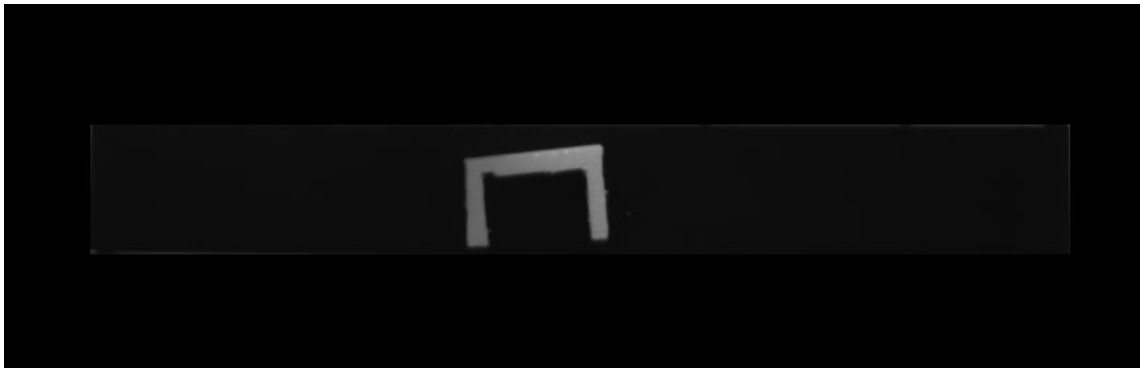

Figure 13. Segmentation of the region of interest in the image

Next, we apply Gaussian blurring to the image. This technique helps to suppress the high frequency noise present in the image and to reduce the level of detail, as shown in Figure 14. The outcome is a smoother image that helps to detect objects more accurately.

Figure 14. Noise reduction in image via Gaussian blurring

Following the Gaussian blurring, we apply a thresholding technique to the image. In this procedure, the grayscale image is transformed into a binary image, in which the pixel intensities are assigned to the maximum value of 255 (white) if they exceed a certain threshold, or to 0 (black) otherwise. This binary transformation accentuates the contrast between the objects and their surroundings, a change that is clearly illustrated in Figure A. Given that the conveyor belt is black, it blends in perfectly with the previously applied black frames. Therefore, setting a threshold relatively close to zero (absolute black) facilitates a remarkable distinction between the objects and the conveyor belt, ensuring accurate identification of the objects in subsequent analysis steps.

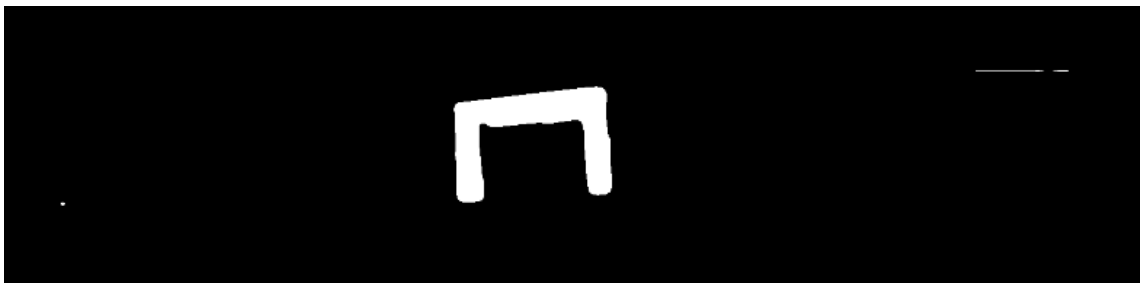

Figure 15. Applying threshold technique for improved object segmentation

Our preprocessing stage concludes with the application of morphological operations—namely erosion and dilation—to the image. These two techniques work hand in hand, with the aim of minimising image noise while retaining as much of the original object detail as possible.

First, we perform the erosion operation. Erosion essentially works by shrinking the objects in the image, thereby reducing their boundaries. This operation helps to eliminate small scale noise. The image post-erosion provides a clearer focus on the main objects, as can be seen in Figure 16.

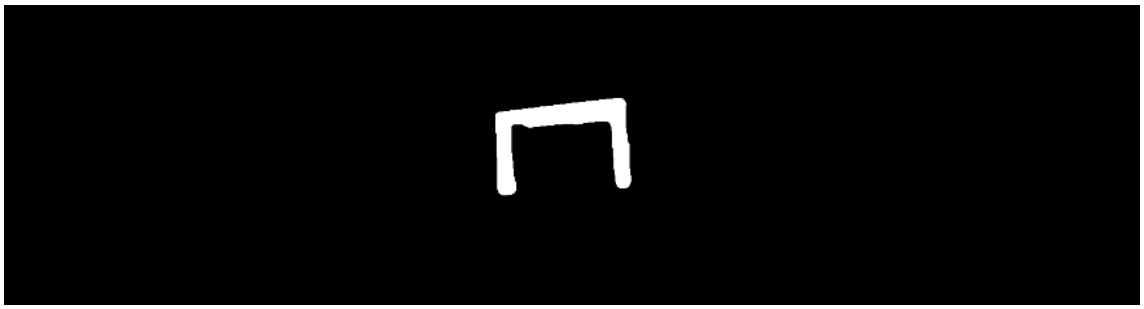

Figure 16. Shrinking object boundaries via erosion operation for noise reduction

Subsequently, we apply the dilation process. Dilation attempts to enlarge the remaining shrunken objects in the image back to their original size. The purpose of this step is to recover the necessary object details lost during the erosion process. The outcome of the dilation process can be seen in Figure 17. This sequence of erosion followed by dilation results in a cleaner and sharper representation of the objects to be manipulated, now with practically no noise.

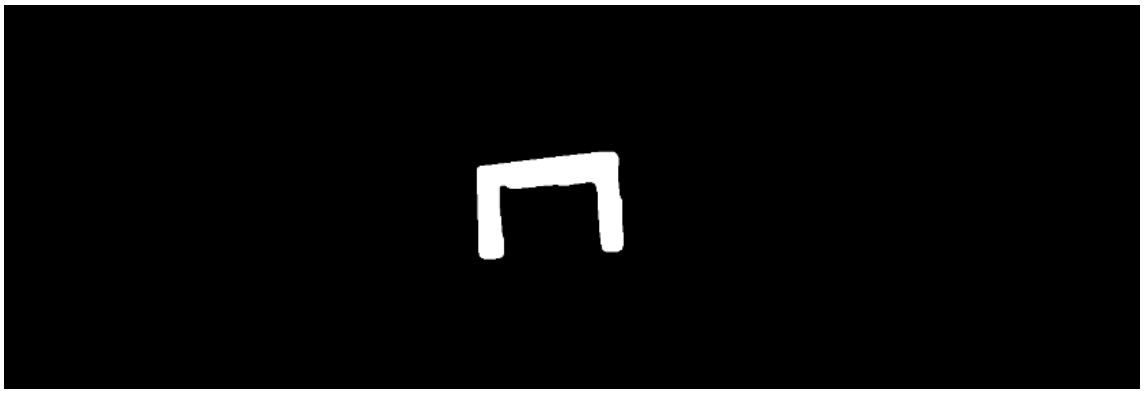

Figure 17. Restoration of object size and details with dilation

In conclusion, our image preprocessing stage involves a series of transformations - masking, blurring, thresholding, erosion and dilation – to facilitate the subsequent stages of object detection, optimal grasping point calculation and stability prediction. Thus, these preliminary transformations lay the foundation for the smooth and efficient execution of the forthcoming steps in the process pipeline.

4.2. Object Detection and Differentiation

In this section, we delve into the next critical phase of our process: Object Detection and Differentiation. This stage essentially picks up where the preprocessing left off. The cleaner, sharper image generated from the preprocessing stage is now subjected to further analysis to identify and distinguish the various objects present in the image. The object detection process, which consists of different sub-steps such as contour detection, filtering, and object centre calculation, is detailed below.

After preprocessing, the image is ready to undergo the process of contour detection. Here, contours are essential mathematical concepts used to represent the shapes of the objects present in the

image. In computer vision, contours are simply continuous curves that join all the contiguous points along the object's boundary that have the same colour or intensity.

We first apply the cv2.findContours function from the OpenCV library to our pre-processed image. This function detects contours in the image and returns a list, with each element of the list representing a unique contour in the image, as a collection of the (x, y) coordinates of the boundary points of the object.

Next, we filter the detected contours based on the number of points they contain. Contours with a minimum of 100 points are retained, and the rest are disregarded. This filtering process helps us in discarding noise and focusing on the significant objects present in the image. The filtered contours are shown in Figure 18.

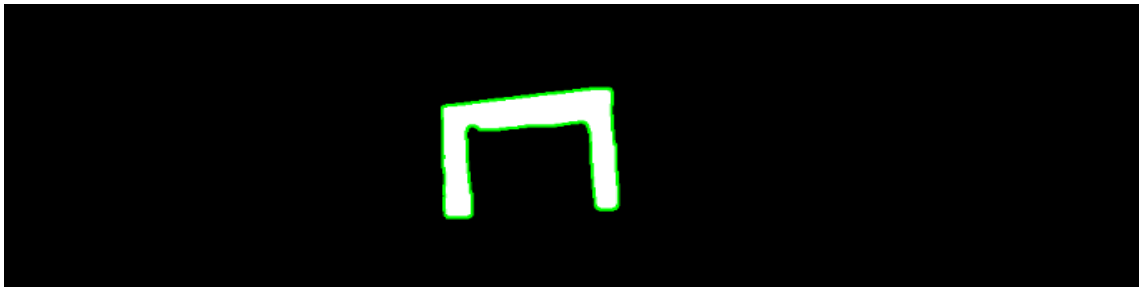

Figure 18. Detection and filtering of contours in the pre-processed image

After capturing the filtered contours, our goal is to discern the objects that these contours represent. To do this, we establish two empty lists: one to store the contours of the recognised objects ("objects") and one to record the corresponding centres ("centres") of these objects/contours.

Then, we calculate the centre of each filtered contour and evaluate if this contour corresponds to an existing object in the "objects" list by checking if the centre of the contour is inside the other contours found thanks to the function cv2.pointPolygonTest. If yes, the contour is appended to this object element (now, object element “i” in the list will have 2 contours, i.e., the object will have 2 contours). If not, we designate this contour as a new object, thus adding it to the "objects" list as a new element, and we enter its centre in the respective element of the "centres" list.

This step mainly allows us to distinguish between objects of 2 different shapes. Objects with an "O" shape (as they will have two contours, both belonging to the same object) and other objects with any other kind of shapes (as they will have only one contour). Thus, by evaluating whether the centre of one contour lies inside another contour, we can distinguish between these two types of objects, as explained above. This differentiation will be very important for calculating the optimal grasping point of any object, since although all objects will use the same method as a basis, for "O"-shaped objects it will follow a slightly different approach.

Below are two images side by side, where we can see the two types of objects that we are going to differentiate as we have explained. Thus, in Figure 19 (representing most of the objects) we can

see how we will only have 1 contour, while in the case of Figure 20 (O-shaped objects only), we will instead have 2 contours for a single object.

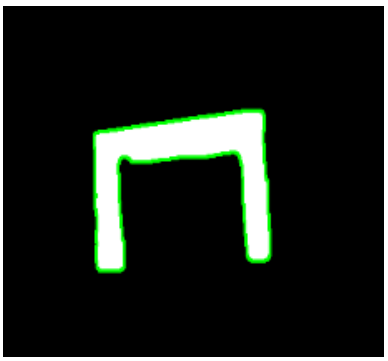

Figure 19. Single contour objects (most objects)

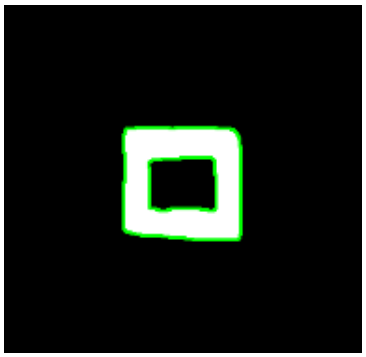

Figure 20. Double contour objects (O-shaped objects)

Finally, to conclude this stage, we select the object with the smallest x-coordinate centre of all detected objects, i.e., the object furthest to the left of the conveyor belt, which will serve as the input for the subsequent step of the process flow, the calculation of the optimal grasping point. Therefore, if there are more objects on the conveyor belt, these will not be considered for the time being, as we can only pick up one object with the robotic arm at a time and calculating the best grasping point of all objects would entail unnecessary processing time. Once the manipulation of an object will finish, the entire process flow will start all over again, giving way to the grasping of the next object.

4.3. Calculation of the Optimal Grasping Point

Following the process flow, we will now delve into our novel method for determining the most suitable grasping location for different types of objects. As noted in the previous section, a critical aspect of our method is the differentiation between O-shaped objects and other shapes. This distinction is due to the fact that O-shaped objects are defined by two contours (one outer and one inner), while all other objects are characterised by a single contour.

The first part of this subchapter will be centred in to elaborating the method we have devised for objects with a single contour. We will detail the calculations, algorithms and strategies used to calculate the optimal grasping point for these items. Following this, we will explain the minor adaptations necessary to apply our method to O-shaped objects, accounting for their unique two-contour structure. This distinction is important because the optimal grasp position for an O-shaped object will use a slightly different approach to accurately compute the most effective grasping location.

4.3.1. Single Contour Objects

For objects with a single contour, the best grasping point is determined by examining each pair of points with equal y-coordinates along the x-axis of the same contour. Below, Figure 21 shows

a visual representation of our method of examining each pair of points under consideration. This image shows our method of going through each y-coordinate of the contour (in green) along the x-axis (lines in red), while highlighting the pairs of points (crosses in yellow) on each of the red lines depicted (only a few red lines are shown for better understanding and visibility). Each of these pair of points are the ones that will be considered in the subsequent calculations.

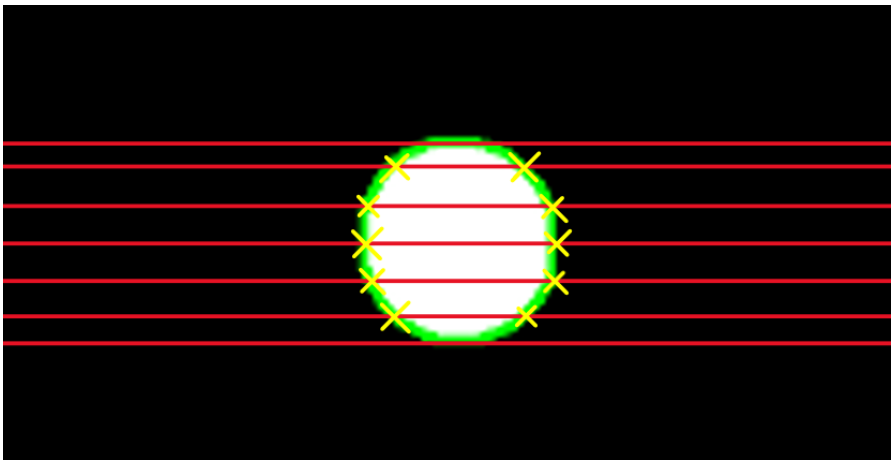

Figure 21. Pairs of points along the x-axis for future evaluation on a single contour object

In the above image, as observed, we also filter out the y-coordinates that have an excess of contour points (more than 15 pixels), thus no pair of points is visible in the top and bottom red lines. This is because a large number of points at a single y-coordinate typically signifies that these points are likely representing the same side of an object. As such, it would not be feasible to achieve any grasping at these locations.

Furthermore, we also ensure that there is a minimum distance of 10 pixels and a maximum distance of 100 pixels between pairs of points. The minimum gap guarantees that the points belong to different sides of the object, which allows for the possibility of grasping the object, while the maximum distance restricts the points that will be considered due to the physical limitation of the maximum opening of the gripper's fingers, thus preventing picking up objects through a pair of points separated by more than a certain distance.

Afterwards, to determine the quality of these selected point pairs, three main measures are considered and scored between 0 (not very favourable pair of points) and 1 (very favourable pair of points for grasping):

1. **Slope Difference:** This measure assesses how parallel the two surfaces on which the two points lie are to each other. The more parallel they are, the more desirable the grasping point and therefore the higher the score. This is because the gripper used is a two-finger parallel gripper, so gripping two points whose tangent lines (slopes of the points) are not parallel is physically more complicated. Thus, if the slopes are perfectly parallel (0 degrees difference), the score is 1 and the less parallel they are the less they score, until the score

reaches 0 when they are completely perpendicular (90 degrees difference). This calculation would be given by the following expression:

$$\boldsymbol{Score_{slope\ dif.}} = 1 - \frac{|slope\ difference|}{90} \tag{11}$$

For the avoidance of doubt, the slopes at the points have been calculated simply by the following formula:

$$\boldsymbol{Slope_{contour\ point\ (x_i, y_i)}} = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) \tag{12}$$

Where ∆x and ∆y are the difference of the value of the variable x and y, respectively, of a point of the contour situated 5 points ahead with another point of the contour situated 5 points before:

$$\boldsymbol{\Delta x_{contour\ point\ (x_i, y_i)}} = x_{i+5} - x_{i-5} \tag{13}$$

$$\boldsymbol{\Delta y_{contour\ point\ (x_i, y_i)}} = y_{i+5} - y_{i-5} \tag{14}$$

The choice of taking 2 points equally spaced by 10 points between them (5 points before and 5 ahead) is the point of equilibrium to obtain a proper calculation of the slope of the tangent line to the contour point in question, because if we take a smaller number the slope could take extreme values and if we take a larger number, we would no longer be considering the correct part of the contour in question.

2. **Distance to Centre:** This metric calculates the distance between the midpoint of the two points and the centre of the contour. The closer this midpoint is to the centre of the object, the better the grasping point, and hence, the higher the score will be. In this case, the formula that can be used to range the score from 0 to 1 will be the following:

$$\boldsymbol{Score_{dist.\ to\ centre}} = 1 - \frac{dist.to\ centre}{max.dist.to\ centre} \tag{15}$$

As it can be seen in the previous formula, the closer it gets the midpoint to the centre of the object, the closer the distance to the centre will be to 0, thus improving the score more and more until it reaches 1. To avoid confusion, the distance to the centre and the maximum distance to the centre are obtained according to the following calculations:

$$dist.to\ centre = midpoint - contour\_centre \tag{16}$$

$$max.dist.to\ centre = \max\left(contour - contour\_centre\right) \tag{17}$$

Where the *midpoint* is just the mean of point 1 and point 2, the *contour_centre* is the mean of all contour points and the *max.dist.to centre* will be the maximum value from any point of the contour to the centre of it.

3. **Inter-point Distance:** This measure simply considers the distance between the two evaluated points. The smaller the distance, the preferable the grasping point and therefore the higher the score. This is because gripping an object through points closer together tends to have more stability than gripping the same object through points further apart. The score is calculated using the following formula, where pairs of points with smaller distances between them are guaranteed to receive higher scores.:

$$\boldsymbol{Score_{inter-point\ dist.}} = 1 - \frac{point\ dist.}{max.\ possible\ dist.} \tag{18}$$

In the above formula, it should be noted that the *max. possible dist.* is determined by the maximum distance in pixels allowed to grasp an object, which, as explained before, corresponds to the maximum separation of the gripper's fingers, taking a value (in pixels) of 100. Thus, any pair of points that are considered to have a greater distance will receive a score of 0 in this section, with a higher score as the assessed points are closer together.

Each of these metrics contributes a weighted portion to the overall score for each pair of points. The weightings are as follows: 50% for the slope difference (w1), 30% for the distance to centre (w2), and 20% for the inter-point distance (w3).

With these weightings, the overall score for each pair of points is computed as:

$$\boldsymbol{Score_{pair}} = w1 * Score_{slope\ dif.} + w2 * Score_{dist.\ to\ centre} + w3 * Score_{inter-point\ dist.} \tag{19}$$

As each pair of points is evaluated, we will overwrite another variable called “best_score” in case the score of the evaluated points is higher than the one we had so far, keeping at the end the pair of points associated with the best score as well as the corresponding score.

At this point, we have the best pair of points along the x-coordinate axis direction for all possible y-values of the contour. As a graphical display of the best grasping point according to the above calculation on a pair of objects, Figure 22 is shown below, where the best pair of points are shown in blue, their midpoint in red, and the tangent lines at each point of the contour in green.

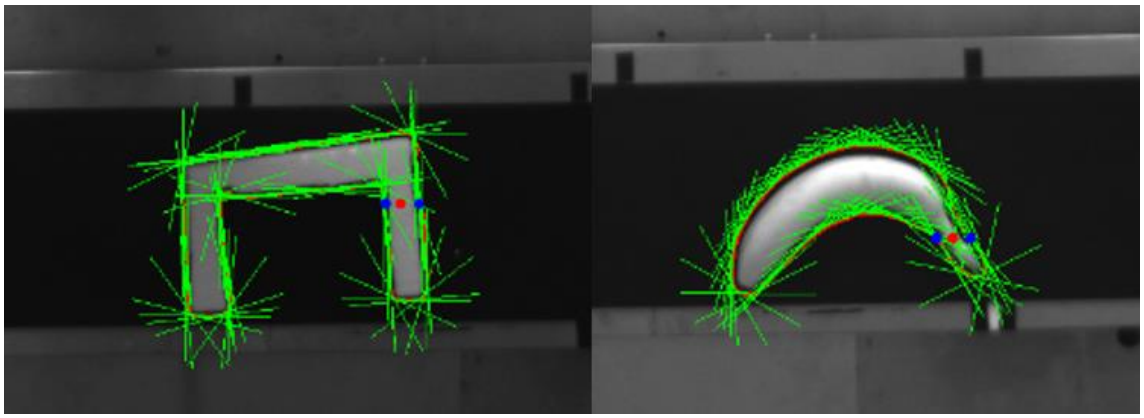

Figure 22. Best pair of grasping points along the x-direction for all y-values of the contour

However, as can be seen these cannot be the best points, as we have only checked pairs of points in one direction, which is along the x-axis.

If we now repeat the above set of calculations, for different directions (at different angles to the x-axis), we can now check which would be the best overall grasping point in any orientation of the end effector. To illustrate this idea, below is a figure showing 2 other directions, that will now be considered among other ones.

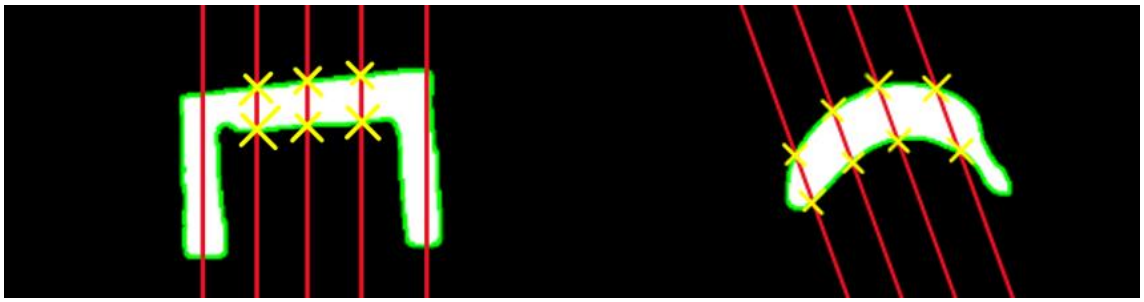

Figure 23. Visualisation of different directions to determine the best overall grasping point

To achieve this, we simply rotate the contour of the object we were working with, so that once the contour is rotated, we recalculate the best point along the x-direction. A better representation of this can be seen in Figure 24:

Figure 24. Visualisation of the actual contour rotation performed to simulate the search for the best pair of points in different orientations

Lastly, we employ the method explained above that performs the three different calculations, obtaining the best score and the corresponding pair of points for each set rotation. If the score of the best point pair in a new rotation exceeds the current best score, we overwrite the variables corresponding to the former best score, the corresponding point pair and the rotation applied. After performing all the scheduled rotations, we have identified the best pair of points in all rotations. These points are then rotated back to plot them on the original unrotated image, and the gripper, when picking up the object, will rotate according to the rotation of the contour that produced the best pair of points.

The rotation to be performed will range from 0 to 180 degrees because angles from 180 to 360 degrees will produce essentially the same contour analysis. To achieve this we use the *np.linspace* function, which generates a sequence of equally spaced values within a defined range. In our case, it will distribute the total degrees of rotation (from 0 to 180) in equal divisions.

A practical trade-off between analysis granularity and computational time is to divide the rotation range into 10 equal parts, resulting in rotations at 20-degree intervals. Therefore, we will run the previous method (with the 3 different calculations) a total of 9 times, as the results for 0 and 180 degrees would be identical and therefore there is no need to compute the 180-degree rotation. This approach allows us to perform the rotation process without requiring excessive computational resources, with the complete cycle of nine rotations taking just around 0,1 to 0,2 seconds.

In short, what we will basically do is rotate the object's contour a total of 9 times, evaluating for each of the rotations the best pair of grasping points. Figure 25 shows a visual representation of the 9 equivalent search directions (red lines) that would be considered the same as rotating the object contour and then searching along the x-axis.

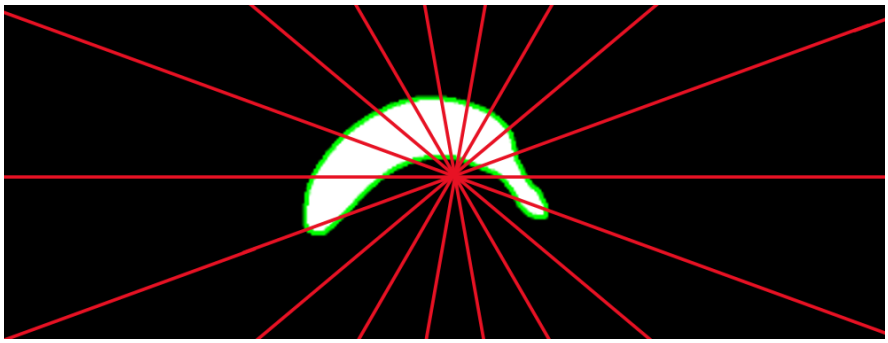

Figure 25. Display of all directions considered for the proper determination of the best pair of points overall

To illustrate that when we look for the best pair of points in all the directions considered, we end up getting the most optimal grasping point possible, Figure 26 below shows that indeed the pair of points obtained now are much better candidates to be grasped from than those obtained previously in Figure 22.

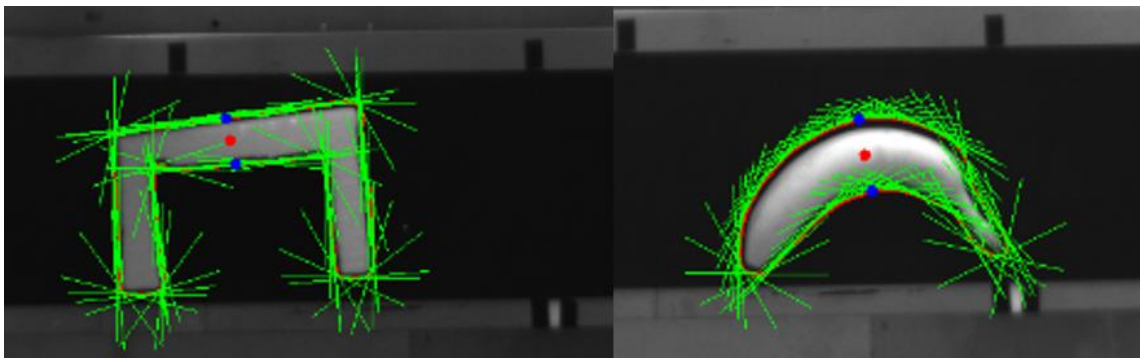

Figure 26. Optimal grasping points across all considered directions for single contour objects

This method provides a highly efficient and accurate evaluation of all possible pairs of grasping points, smartly bypassing numerous iterative computations in areas that would not significantly enhance the result, thereby saving substantial computational time without compromising the quality of the output.

Furthermore, to ensure the reliability of the identified best pair of points, before sending to the robot arm the corresponding command to grasp the object, we verify that the same best pair of points is consistently retrieved at least 15 times. This precautionary step eliminates the risk of premature command execution, which could potentially occur from a rushed object detection.

4.3.2. Double Contour Objects

In the case of two-contour objects, the process to be followed and the calculations to be performed are exactly the same, except that now the pair of points to be evaluated must each belong to one of the 2 contours. A graphical view of these possible points to be taken into account, similar to the previous case with single contour objects in Figure 21, is shown in the following image:

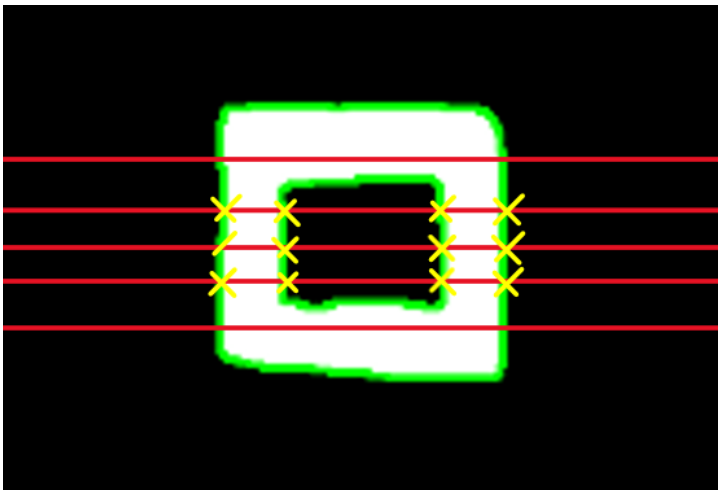

Figure 27. Pairs of points along the x-axis for future evaluation on a double contour object

The only additional modification for dual-contour objects pertains to the assignment of weights to our evaluation metrics. In the case of single contour objects, the weights allocated were 50% for the slope difference (w1), 30% for the distance to the centre (w2), and 20% for the inter-point distance (w3). For dual-contour objects, however, these weightings are adjusted to 40% for the slope difference (w1), 40% for the distance to the centre (w2), and 20% for the inter-point distance (w3), providing a more balanced influence of the difference in slope and the distance to the centre on the final score. This is because usually the slope of the 2 points of the 2 different contours, will in most cases already be parallel with each other, so the weighting of the slope difference will not be as important as before. On the other hand, increasing the weight of the distance to the centre will enable us to get the midpoint of the sides of polygons in the form of squares, triangles and others.

To clearly illustrate the result obtained with the slightly modified method for determining the optimal grasping pair of points for objects with two contours, we present the following picture:

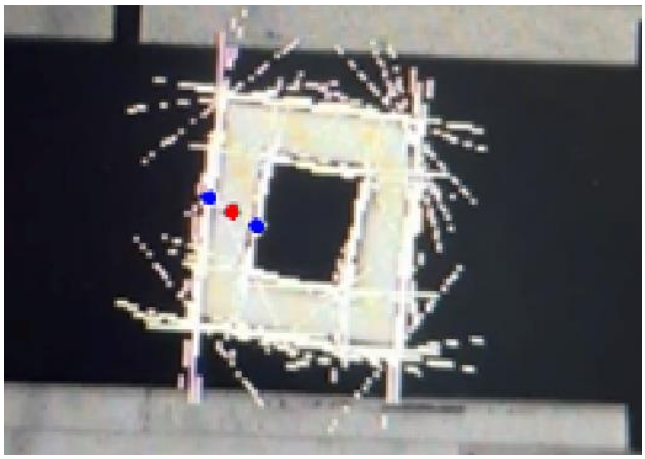

Figure 28. Optimal grasping points across all considered directions for double contour objects

As can be seen in the image, the selected pair of points (in blue) is a pair of points each belonging to one of the contours. We can also see how the midpoint of these 2 points (point in red) is the closest possible point to the centre of the object.

Having explained our 2 methods of calculating the best grasping point adapted to the different types of objects, we will illustrate the sequence of events in the process. This includes from image pre-processing to the transmission of the calculated optimal grasping point to the grasping stability predictor. The details of this predictor will be explained in the next section.

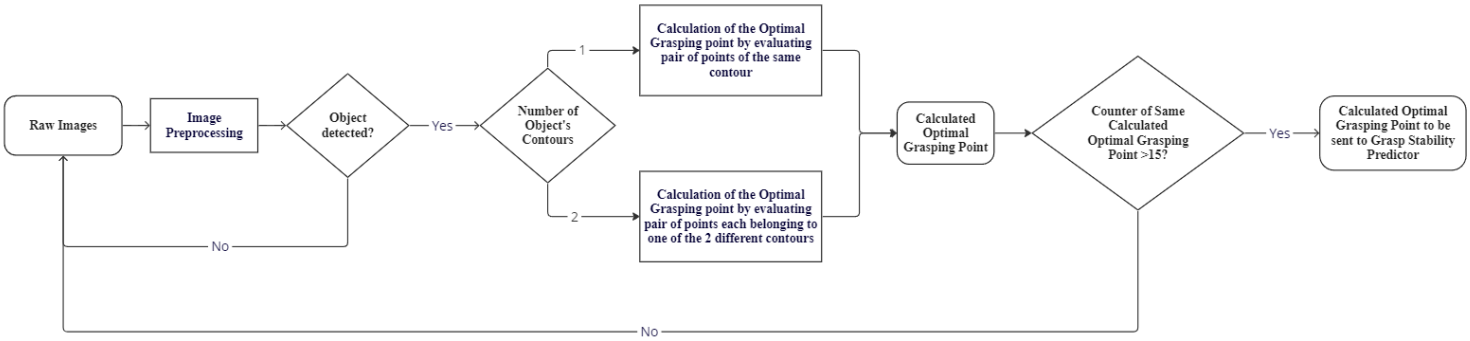

Figure 29. Flowchart of events following Optimal Grasping Point Identification

4.4. Grasp Stability Prediction using Machine Learning Techniques

In this section, we outline the process of implementing Machine Learning techniques, such as Decision Tree Classifiers introduced in Section 2.3, to predict the stability of the grasping task. Stable object grasping can be defined as the continuous ability to pick up, hold, and manipulate an object without accidentally dropping it. The purpose of this endeavour is to proactively prevent unsuccessful grasping attempts by predicting their likelihood of success and failure using a Decision Tree Classifier model previously trained on historical data.

4.4.1. Dataset and Features

The dataset used in this project is composed of 148 records, each documenting an individual grasping attempt. These records were collected through extensive experimentation involving objects of various shapes and sizes and attempts to grasp these objects from different positions and rotations.

Each record consists of three key attributes and their associated grasp attempt result. The 3 attributes are the X and Y coordinates, representing the grasping position, and Oz, which indicates the grasping rotation. The grasping position provides the spatial location of the object in relation to the robot (in metres), while the grasping rotation provides the angular orientation of the gripper during the grasp attempt (in radians). These attributes are obtained from the Calculation of the Optimal Grasping Point explained in detail in the previous section, but not after a small transformation that will be explained in detail in the upcoming Section 4.5. Robot Commands Execution. This basically done to convert the calculated grasping position from the image, from pixels to metres, as this is the unit that we will use to send the robot commands to grab the object. Lastly, the result of each attempt, designated as either a success or failure, serves as the binary output for each record.

4.4.2. Model Training and Testing

The training and testing of the Decision Tree Classifier model are conducted using the scikit-learn library in Python [19]. Initially, the dataset is partitioned into a training set and a test set, where 70% of the dataset's records are randomly selected for training the model, and the remaining 30% is reserved for testing. This division ensures that the model has a broad and representative learning base while maintaining a substantial subset of data to independently validate the model's effectiveness.

Following this, the Decision Tree Classifier model is initialised without any pre-set parameters. This allows for the implementation of the GridSearchCV function, a hyperparameter tuning technique that examines a wide range of potential configurations. The GridSearchCV function essentially builds and evaluates a model for every combination of the specified parameters, such as maximum tree depth, minimum sample splits, and minimum samples at leaf nodes.

In this specific project, the tree depth was explored within the range of 2 to 4, the minimum samples required to split a node ranged from 2 to 10, and the minimum samples required at a leaf node varied from 4 to 15. By searching over these defined parameters, GridSearchCV helps identify the combination that yields the best performance, providing the most suitable parameters for the Decision Tree Classifier model.

To evaluate the performance of the decision tree classifier model, a set of performance metrics is used to provide a comprehensive assessment. The main metric used is accuracy, which gives a measure of the overall correctness of the model's predictions. However, given the unbalanced distribution of classes in our dataset, where there are more successes than failures in grasping

attempts, relying on accuracy alone may give a misleading interpretation of the model's performance. To counter this, additional metrics such as precision, recall and F1Score are used.

- **Accuracy:** Accuracy measures the overall correctness of the model's predictions, given by the ratio of true predictions (both positives and negatives) to the total number of predictions.
- **Precision:** Precision quantifies the model's exactness, given by the ratio of true positive predictions to all positive predictions (i.e., from the true predictions, how many are actually true?). This metric provides insight into the reliability of the model's positive predictions.
- **Recall:** Recall, also known as sensitivity or true positive rate, assesses the model's ability to identify all actual positive instances (i.e., from all the actually true cases, how many were predicted also as true?). It evaluates the model's ability to avoid false negatives, reflecting its completeness.
- **F1 Score:** The F1 score is the harmonic mean of precision and recall, providing a balanced measure between the two metrics. This approach is especially useful when you want to evaluate the performance on imbalanced datasets, which is our case.

Once the performance of the model has been evaluated and optimal parameters identified through iteration with GridSearchCV, these parameters are used to train the decision tree classifier model. The model, trained with these optimal parameters, is then used to predict outcomes on test data not seen during the training phase. This methodology allows for a rigorous and independent evaluation of the model's ability to accurately generalise and predict outcomes, thus assessing its practical applicability.

4.4.3. Model Visualisation

The trained model is visualised using the plot_tree function from the scikit-learn library. This graphical illustration provides detailed overview of the decision-making process of the model. Each decision node in the tree signifies a decision based on the attributes (grasping position and rotation), leading to a leaf node that represents either a success or failure outcome.

Note: Further analysis, including a discussion on the model's accuracy and the resultant implications, will be presented in Chapter 5 (Results). The purpose of this section is to detail the implementation process of the Decision Tree Classifier for predicting grasp stability.

4.5. Robot Commands Execution

The final stage in our process flow, prior to the actual task of object manipulation, is sending the necessary robot commands to the robot. This critical phase involves transforming theCalculation of the Optimal Grasping Point, identified in Section 4.3, from pixel coordinates (in the processed

image), to the physical distance from the robot to the intended grasping point on the object (in meters).

The transformation from image-based coordinates to real-world measurements is a vital bridge between computational analysis and practical execution. This conversion feeds the Inverse Kinematics solver with tangible world-coordinate values, that, together with the controller, will drive the robotic gripper to the desired position for proper object manipulation.

Furthermore, the use of physical measurements enhances our understanding of the objects' placement in the real world. Lastly, it is important to notice that these measures are the ones that were also used when predicting Grasp Stability Prediction using Machine Learning Techniques, as explained in Section 4.4 earlier. In essence, this final process in our pipeline ensures that the robot can effectively act on the processed information, thereby efficiently performing the object manipulation task.

4.5.1. Translation from Pixel Coordinates to Real-World Positions

For a clearer perspective, we will first examine an example of an image captured by the camera, zoomed to the region of interest (Figure 12 has already shown a complete view of the raw image). This time, in this image, we also have added reference points and their corresponding coordinate axes, as well as some numerical measurements. All this will be analysed and explained in detail below, with the aim that this contextual visualisation will be of great help to understand the next transformation process.

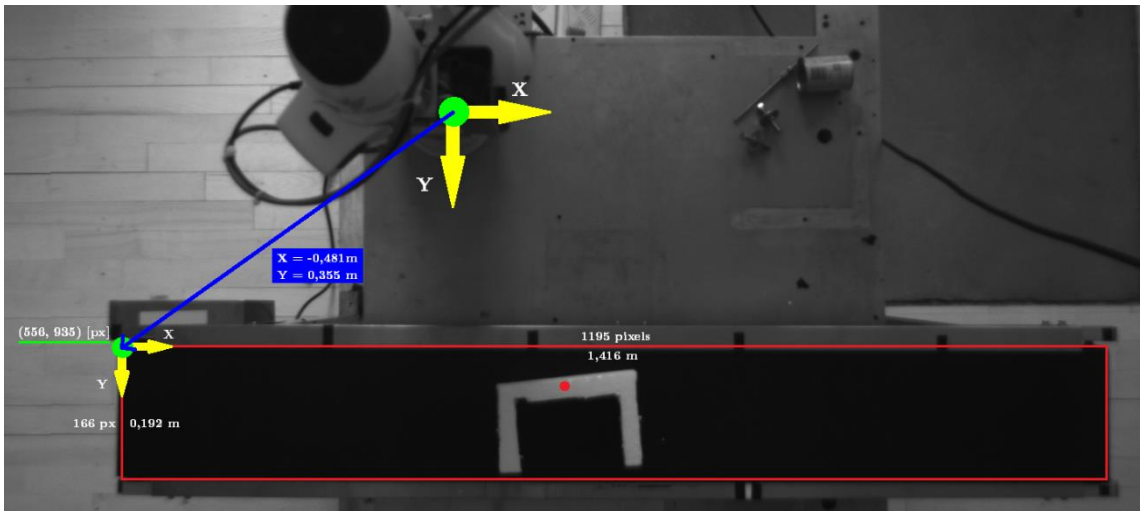

Figure 30. Zoomed raw camera image with reference points, axes and measurements

Ultimately, our objective is to transpose the optimal grasping position (depicted as the red point on the object in Figure 30) into the real-world coordinates relative to the robot's frame of reference. As an initial step, we will reference the grasping point (P) to the top-left corner of the conveyor belt, which, on the full-size image, holds coordinates x = 556 and y = 935:

$$\boldsymbol{P_{x_{conv.\_belt\_reference}}}\,[\boldsymbol{px}] = P_{x_{full-size\ image}}[px] - 556 \tag{20}$$

$$\boldsymbol{P_{y_{conv.\_belt\_reference}}}\,[\boldsymbol{px}] = P_{y_{full-size\ image}}[px] - 935 \tag{21}$$

Afterwards, we can convert the units of the position of the grasping point from pixels to meters. This transformation is possible by determining the size of the conveyor belt in both pixels and meters, as indicated by the red rectangle in Figure 30:

$$\boldsymbol{P_{x_{conv.\_belt\_reference}}}\,[\boldsymbol{m}] = P_{x_{conv.\_belt\_reference}}\,[px] \cdot \frac{1{,}416\ [m]}{1195\ [px]} \tag{22}$$

$$\boldsymbol{P_{y_{conv.\_belt\_reference}}}\,[\boldsymbol{m}] = P_{y_{conv.\_belt\_reference}}\,[px] \cdot \frac{0{,}192\ [m]}{166\ [px]} \tag{23}$$

Finally, the last transformation consists of shifting the reference of the grasping point from the top left corner of the conveyor belt to the centre of the robot base, where its coordinate origin is located:

$$\boldsymbol{P_{X_{robot\_reference}}}\,[\boldsymbol{m}] = P_{x_{conv.\_belt\_reference}}\,[m] - 0{,}481 \tag{24}$$

$$\boldsymbol{P_{Y_{robot\_reference}}}\,[\boldsymbol{m}] = P_{y_{conv.\_belt\_reference}}\,[m] + 0{,}355 \tag{25}$$

4.5.2. Sequence of Events following Grasping Point Identification

With the grasping position now defined in suitable units, we could then move on to the Grasp Stability Prediction using Machine Learning Techniques, as detailed in Section 4.4. The result of inputting this position into the trained model will be the probability of successful grasping. If this probability is too low (less than 50%, for instance), the process flow would jump back with a warning to the Calculation of the Optimal Grasping Point step. Then, we could either change the position of the current object to a section of the conveyor belt that is known to have a higher probability of success (known thanks to the evaluation of the results from the Decision Tree Classifier that will be seen in the next chapter) or see if the new calculated best grasping point has a higher probability of grasping success (although we are likely to stand at the same point as the best calculated point will not deviate much). This integration of stability prediction increases the flexibility and decision-making capability of our robotic system, avoiding wasting time on attempts that are likely to fail.

If we proceed with the grasping attempt, the robot arm will move to the calculated position, just above the object, and close the gripper. It is now when we make use of the gripper's tactile sensors to evaluate whether an object has been grasped correctly, stopping the procedure again if it has not. This evaluation is basically done by setting a certain threshold on the pressure received by the sensors, which if not surpassed means that the object has not been gripped firmly enough.

Finally, at this point, all that remains is to manipulate the object by moving it from the conveyor belt to a designated area to release the objects. The following figure shows a flowchart of all the processes and evaluations performed as explained in detail in this subsection.

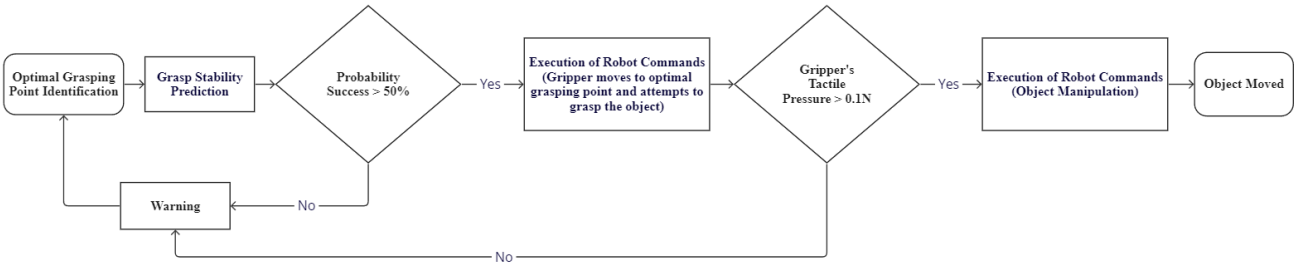

Figure 31. Flowchart of events following Optimal Grasping Point Identification

5. Results

This chapter will present the concrete findings from our testing process. To effectively quantify these results, our evaluation is mainly based on visual comparisons of the calculated grasping points, as well as the measurement of the "success rate" (our main numerical KPI), to objectively measure the efficiency of our developed system. Together, these methods enable us to examine the benefits of our work in a comprehensive and methodical manner.

5.1. Grasping Point Identification

In this section we will compare our newly devised method for the Calculation of the Optimal Grasping Point of an object, against Meile's approach, explained in depth in Section 2.2.3. As already mentioned, it will be of particular interest to see how an overly simplistic approach can give good results with objects of simple shapes (in the concrete case of Meile's method, only with circular or rectangular shapes), but on the other hand have difficulties with objects of other shapes. For this reason, although we have already shown some results of our new method in section 4.3, this time we will show side-by-side images of both methods for calculating the grasping point for better comparison and evaluation.

5.1.1. Evaluation of Grasping Points on Simple-Shaped Objects

We will now evaluate the grasping point obtained with each of the 2 methods on circular and rectangular objects (simple-shaped objects). In the following figures, on the left side of the image we will be able to see the grasping point obtained with Meile's method while on the right side, the one obtained with our new method.

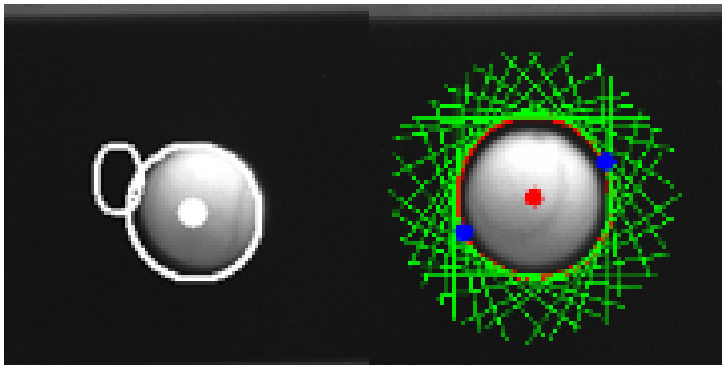

Figure 32. Tennis ball grasping point comparison: Meile's method (left) vs Novel method (right)

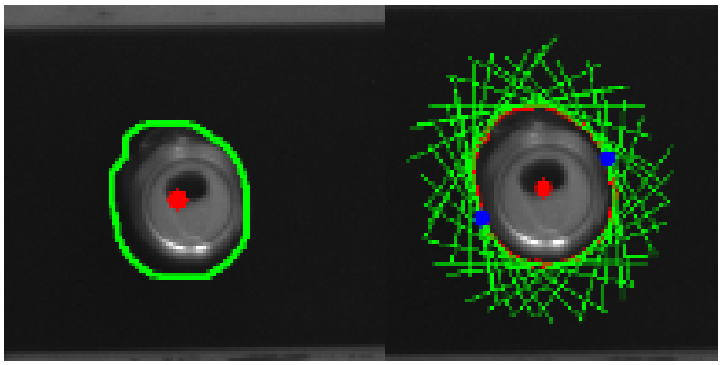

Figure 33. Metal can grasping point comparison:
Meile's method (left) vs Novel method (right)

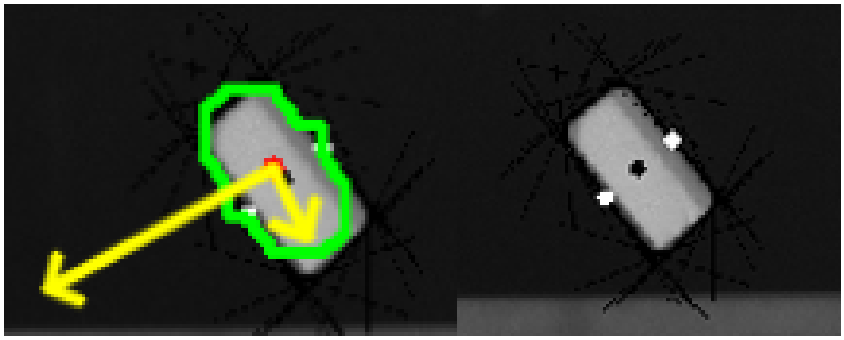

Figure 34. Wood block grasping point comparison:
Meile's method (left) vs Novel method (right)

As anticipated, for objects with simple geometric shapes, both the proposed method and Meile's approach deliver effective and similar good grasping points. This result, in the case of Meile's method, can be attributed to the fact that the calculation of the centre and principal orientation (via the PCA), in simple shaped objects such as circles or rectangles, is a good enough method and results in good grasping points. In our case, the newly devised method also demonstrates a good calculation of the grasping points for such objects, as can be seen in the images above. Consequently, both methods demonstrate similar performance in these scenarios, identifying the optimal grasping points that are likely to result in a successful grasp.

5.1.2. Evaluation of Grasping Points on Complex-Shaped Objects

Building on the results obtained from the simple-shaped objects, we will now proceed to a more complex scenario: objects with intricate shapes. This evaluation aims to challenge both methods and explore the grasp point outcomes on shapes that are far from the standardized geometries we previously examined.

In the upcoming figures, we maintain the same comparison format. Meile's method results are depicted on the left side, while our method's results are on the right side. This direct juxtaposition allows for an immediate visual comparison between the calculated grasp points.

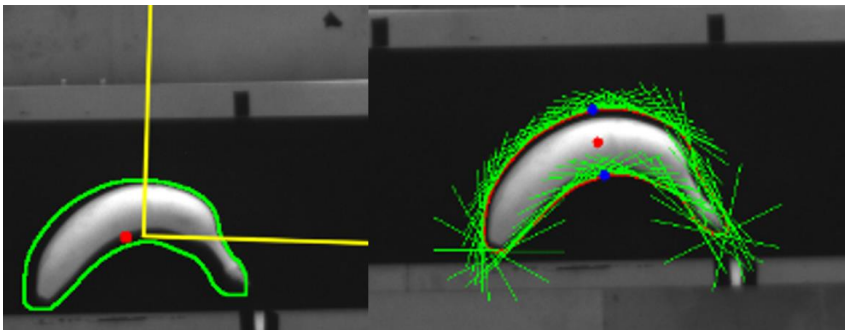

Figure 35. Banana grasping point comparison: Meile's method (left) vs Novel method (right)

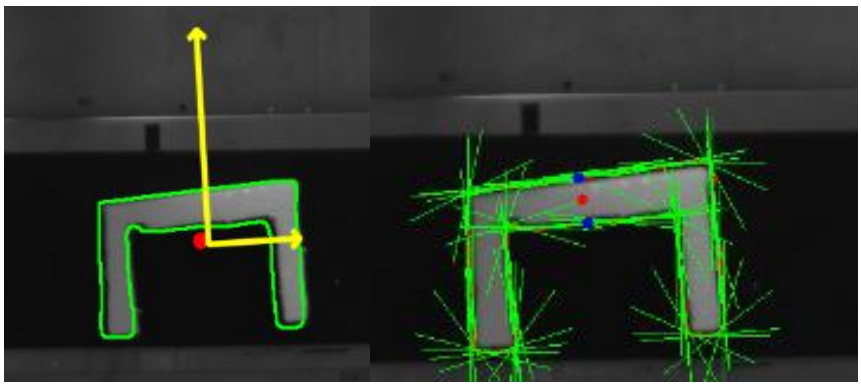

Figure 36. Complex-shaped object grasping point comparison: Meile's method (left) vs Novel method (right)

Figure 37. O-shaped object grasping point comparison: Meile's method (left) vs Novel method (right)

As it can be seen in the previous images, the complex shapes of the objects under consideration in this section pose a significant challenge, as the optimal grasping points are not as straightforward or easily determinable as they were with simpler shapes. Thus, the more complex object shapes examined in this section showcase a noticeable divergence in the performance of Meile's method and our newly devised method.

With these intricate shapes, the underlying assumptions of Meile's method, taking the centre of the object as the grasping position and the main object orientation, with PCA, as the gripper orientation, are less applicable, leading to varying degrees of effectiveness in identifying optimal grasping points.

In some instances, Meile's method's calculated grasping point comes reasonably close to the optimal one, such as in the case of the banana. However, in more challenging cases like the O-shaped object, the calculated point is significantly off from the actual optimal position.

To underscore the implications of minor deviations in determining the optimal grasping point, let's consider one of Meile's experiments involving the task of grasping a banana. The success rate in this case, in his experiments, was only 18%, significantly lower than the average success rate of 86% that he achieved with the simple-shaped objects (excluding IK failure attempts). This discrepancy clearly demonstrates that the accuracy of determining the optimal grasping point influences the overall efficiency and success of the task. In our case, the success rate of grasping the banana was even better than the 86% average of his other simpler objects, with a success rate of around 90%, let alone the low 18% in the case of his experiments with the banana. Furthermore, it should be noted that in our case, when we had an unsuccessful attempt to manipulate the banana, it was not because the optimum point was not correctly found, which in fact was done correctly 100% of the time, but because the gripper did not close correctly for some reason on that occasion.

In conclusion, this section highlights the limitations of Meile's approach when dealing with complex object shapes. The Meile method's reliance on the object's geometric centre and its principal orientation is not adequate for accurately determining the optimal grasping point and orientation. The observations made in this section underline the need for more sophisticated methodologies that can consistently and accurately identify optimal grasping points on a diverse range of object shapes, such as the one developed in the present project.

5.2. Grasp Stability Prediction

In this section, we will critically examine the decision tree model developed for predicting the stability of a grasp. This model, as previously described in Section 4.4. Grasp Stability Prediction using Machine Learning Techniques, aims to enhance the efficiency of the robotic system by making intelligent choices in grasp attempts. To better comprehend the performance and effectiveness of the model, we will divide our examination into two subsections: Performance of the Decision Tree Classifier Model and Evaluation of the Generated Decision Tree Structure.

5.2.1. Performance of the Decision Tree Classifier Model Obtained

In this subsection, we will explore the overall performance of the classifier. We will investigate the model's predictive power, and evaluate its accuracy, precision, and recall metrics, amongst others. As previously explained in depth in Section 4.4.2, these metrics are crucial for understanding the model's ability to make accurate predictions.

Our model obtained an accuracy score of approximately 73.33%, suggesting that it correctly identifies the result of a grasp attempt in approximately 73% of all cases. While this performance is encouraging, it's crucial to consider this in the context of our dataset's class distribution, which

is unbalanced (76% of Success vs. 24% of Failure). This unbalance can skew the accuracy metric, potentially making it appear more optimistic than it is in reality.

In unbalanced datasets, a high accuracy can sometimes be achieved by a model that simply classifies most instances into the majority class. In this context, our model's accuracy score is in line with the distribution of the majority class (76%), which could imply that the model might be biased towards predicting the majority class.

Hence, while accuracy is a helpful metric, it's critical to also consider other metrics that give a fuller picture of the model's performance, especially in the context of unbalanced datasets. In our case, the precision, recall, and F1 score provide additional important insights into the model's performance. Consequently, it is prudent that we delve into these performance metrics as well:

- The accuracy of our model is 0.75, which indicates a high reliability of successful grasp predictions. When our model predicts a successful grasp attempt, it is right 75% of the time.
- The model's recall, or sensitivity, score is quite high, at 0.9375. This suggests that the model excels at identifying actual successful grasp attempts, accurately predicting 93.75% of all actual successful grasps.
- The F1 score is approximately 0.8333, suggesting a strong balance between accuracy and recall. This score suggests that, even with the unbalanced distribution of classes in our dataset, our model effectively balances between minimizing false positives and identifying true positives.

Upon evaluating all these metrics, it becomes apparent that our decision tree model, while not perfect, does exhibit solid performance. Despite certain challenges inherent to the unbalanced nature of our dataset, the model manages to demonstrate a noteworthy ability to predict grasp stability. This validates the potential of decision tree classifiers in this application, while also suggesting some possible room for improvement.

5.2.2. Evaluation of the Generated Decision Tree Structure

Moving forward, we will closely scrutinize the structure of the decision tree itself. We aim to understand the critical decision nodes, thresholds and branches that play a key role in grasp stability predictions. This will help us appreciate the key features that the model considers vital in determining the success or failure of a grasp attempt. The knowledge of these aspects will assist in enhancing the model's effectiveness and guiding future improvements.

For the generation of our decision tree, we have adopted a grid search strategy to determine the optimal hyperparameters, as explained in depth at 4.4.2. Model Training and Testing. This approach involves an exhaustive searching through a manually specified subset of the hyperparameter space of the decision tree model. In our case, the best parameters, obtained after cross-validation, were as follows:

- Maximum depth of tree: 3
- Minimum samples at leaf node: 5
- Minimum samples required to split an internal node: 2

With these optimal parameters, our decision tree model was trained and tested to make predictions. The decision tree finally generated is shown below.

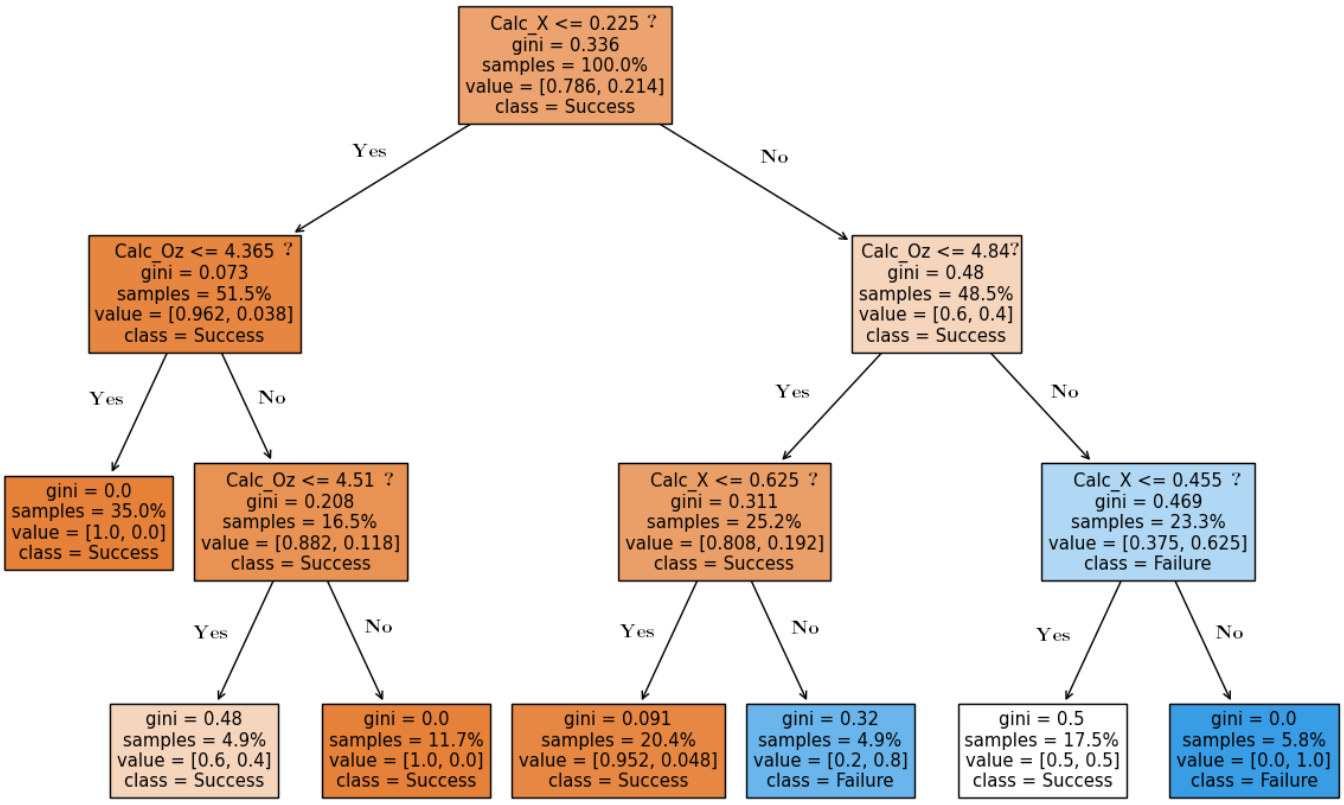

Figure 38. Visualisation of the generated decision tree

The generated Decision Tree visualization represents the model's learned decision rules based on the training data. Each box in the diagram, or 'node', represents a decision based on the values of one of the attributes (Calc_X, Calc_Y and Calc_Oz, representing the X, Y position of the calculated grasping point and the calculated rotation of the gripper respectively) or a final prediction (Success or Failure). Details of each of the measures that can be observed in the different nodes are listed next:

- **Internal Nodes (Decision Nodes):** The internal nodes (not the final leaf nodes) show the decision rule used to split the data. Each decision node contains a condition based on one of the attributes. For instance, on the first node the condition is "Calc_X <= 0.225", so if the value of the feature "Calc_X" is less than or equal to 0.225, the decision tree will follow the left branch, otherwise, it will follow the right branch.

- **Gini:** The Gini index, as explained in 2.3.2. Understanding Gini Impurity, is a metric that quantifies the purity of the node/leaf. A smaller Gini index means that the node is purer, with a Gini index of 0 representing the perfect purity where all elements belong to a single class. For a binary classification problem, the maximum Gini index is 0.5, which represents a node in which elements are equally split into the two classes.

- **Samples:** This field represents the number of samples at each node which satisfy the conditions from the root node (the one on top) to the current node.

- **Value:** This field indicates the distribution of samples of each class at the respective node. It provides the proportion of each target class samples that ended up in that node, in our case, [% Success, % Failure].

- **Class:** This information displays the class that the model predicts for data points that reach that node.

Each path from the root of the tree to a leaf represents a distinct classification decision sequence. Moreover, the decision criteria applied at the higher-level nodes (closer to the root) represent the most influential attributes in determining the outcome of the object grasping attempt.

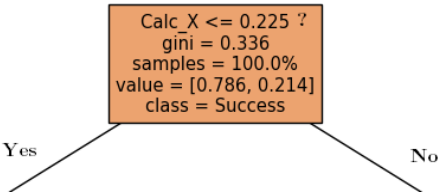

Figure 39. Illustration of the first split in the decision tree

For our decision tree, the initial split is based on the calculated X-coordinate (Calc_X). More specifically, the tree partitions the data by evaluating whether Calc_X is greater than or less than 0.225. This split aligns with our expectations. As explained in section 2.2.2. Inverse Kinematics Failures, we are aware that certain IK issues arise when attempting to grasp objects located in specific areas of the conveyor belt, particularly on the rightmost part of it. A closer examination of the leaf nodes following the first node split confirms this observation.

Notably, from the initial split at X=0.225m, we observe a greater likelihood of “Failures”. The right branches of the tree, which result from the condition X>0.225m, predominantly classify instances as “Failures”. In contrast, the left branches, derived from the condition X<=0.225m, tend to result in “Success” classifications. This distribution reaffirms our understanding that objects placed in the right end of the conveyor belt — more specifically when X>0.225m — are more challenging to grasp successfully. For a more comprehensive understanding, see previous Figure 30, Zoomed raw camera image with reference points, axes and measurements, where it can be seen that the X-axis origin of the conveyor belt is situated at X= -0.481m, and the total length of the conveyor belt extends to 1.416m. Consequently, any point at X>0.225m means any location from around the middle of the conveyor belt towards its right end.

Continuing the analysis of the decision tree, we observe a somewhat surprising trend in the subsequent nodes following the initial split. Both branches resulting from the first split at Calc_X <= 0.225 now focus on the gripper's grasping rotation (Calc_Oz) as the decisive attribute:

Figure 40. Illustration of the subsequent nodes of the first splitting of the decision tree

Interestingly, in both nodes, a similar pattern emerges whenever the Calc_Oz value exceeds approximately pi and a half ($3\pi/2 \approx 4.71$), the corresponding branches lead to subsequent nodes with a higher probability of "Failure". This trend holds true regardless of whether the initial Calc_X value was less than or greater than 0.225m, but much more noticeable when Calc_X > 0.225, since more leaf nodes and percentage of samples results in "Failure".

On the other hand, branches that don't exceed this Calc_Oz threshold tend to lead to nodes with a higher probability of 'Success'. This finding suggests that the gripper's rotation (Calc_Oz) beyond $3\pi/2$ significantly impacts the success rate of the grasping attempt, which in combination with the IK problems mentioned above, makes the probability of successful gripping on the right side of the conveyor belt even lower.

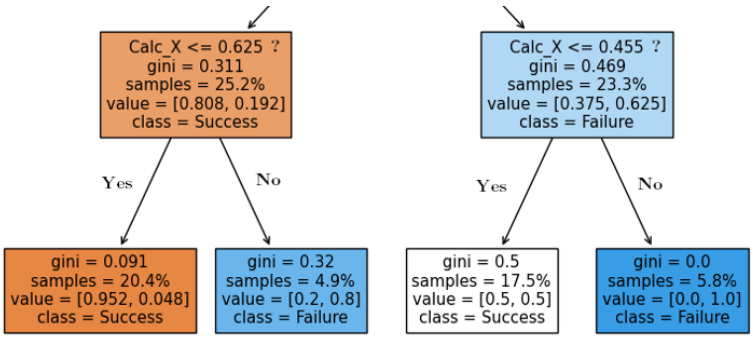

Figure 41. Illustration of the right-hand nodes of the second splitting of the decision tree

A closer examination of the decision tree, in particular the right-hand branches (originating from Calc_X > 0.225), reveals a strong correlation between increasing X values and classification results. In particular, it is evident that "Failure" results become more frequent as X values increase.

Delving deeper into these nodes, it is evident that for X values greater than 0.625, regardless of the Oz value, there is a high probability of a "Failure" outcome. This phenomenon is predominantly attributed to IK failures.

On the other hand, on the rightmost nodes, we notice that for X values greater than 0.455, the effect of Oz being greater than $3\pi/2$ significantly impacts the classification outcome, consistently resulting in "Failures".

In summary, the decision tree reveals a clear trend: as X values increase beyond 0.225, in conjunction with Oz values greater than $3\pi/2$, the likelihood of a "Failure" classification increases substantially. This indicates that the X location of the object and the Oz rotation of the gripper play significant roles in the success of the grasping process.

We cannot do anything about the X location of the object, but we can change the operation of the robot in relation to the gripper orientation. Thus, by maintaining Oz values lower than $3\pi/2$, we can potentially prevent datapoints from branching into the right side of the decision tree, where failures are more likely to occur. This strategy will be a practical way to partly mitigate grasping failures occurring when trying to grasp objects placed on the right part of the conveyor belt. To understand the logic behind this action, it is first necessary to explain how the rotation of the gripper has worked so far.

Initially, the range of rotation for the gripper was set from π to $2\,\pi$, with the gripper's idle state at π. Hence, the rotation considered was solely in one direction (from π to $2\,\pi$). When the grasping task required a rotation larger than $\pi/2$ (leading to final Oz $> 3\pi/2 \approx 4.71$), the gripper would attempt large rotation to achieve the desired rotation rather than a small rotation in the opposite direction that would lead to the same grasping. This approach led to unnecessarily large rotations, and consequently, potential failures in the grasping task when combined with IK disorders.

To address this, we revised the range of rotation now to be from $\pi/2$ to $3\,\pi/2$, effectively still covering all possible orientations as before, but now without the need for such large rotations. Instead of rotations from 0 to $+\pi$ radians, we now perform smaller rotations ranging from $-\pi/2$ to $+\pi/2$.

This modification resulted in better performance, particularly in areas where the robot previously struggled to grasp objects successfully, notably the right part of the conveyor belt where inverse kinematics failures were frequent. Although this adjustment hasn't completely eradicated the problem, especially in areas with very large X values (far right of the conveyor belt), it has considerably reduced the likelihood of such failures.

In conclusion, the use of machine learning methodologies, such as the decision tree classifier, has allowed us to understand the key attributes that lead to lower successful grasping attempts. By identifying and bypassing some of the problematic attribute values, we have therefore been able to improve the overall success rate of our grasping attempts. These findings underscore the invaluable role of advanced machine learning techniques in optimizing the performance and efficiency of systems.

5.3. Overall Grasping Performance

In this section, we aim to replicate and expand upon certain experiments originally conducted by Meile, in order to quantify the overall improvements in grasping performance accomplished through our primary enhancements, previously discussed. These improvements mainly consist of the design of our new method for the Calculation of the Optimal Grasping Point and the mitigation of some of the IK failure attempts by restricting the gripper rotation angle, as explained in the previous section.

In addition to these primary modifications, several smaller incremental improvements have been made to the overall system that are not covered in detail here, yet have cumulatively contributed to significant system enhancement in terms of grasping success rate, speed and stability. By reintroducing Meile's experiments within this evolved framework, we seek to highlight the compound benefits of these improvements on the system's overall grasping performance.

As said, in the interest of drawing comparative insights, we reconstructed the same experiments as in Meile's project. This involved 100 trials across 5 distinct simple-shaped objects (seen in Figure 9), each object being grasped 20 times from four different set areas as per the original experiment. To visualise these 4 different areas to be used for analysis, let's take a look at a very similar image we have seen earlier above, this time detailing the areas of interest above the conveyor belt:

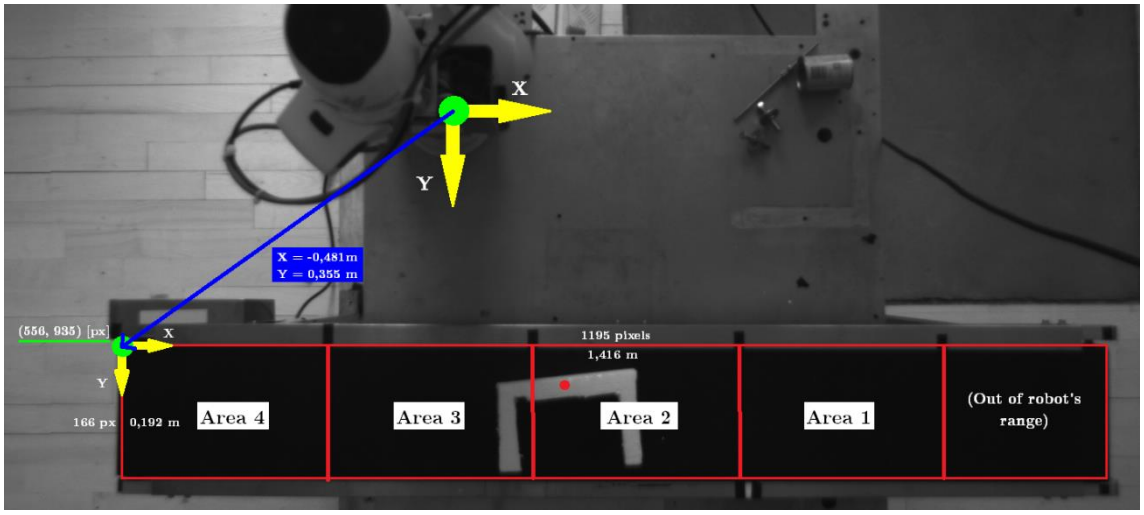

Figure 42. Conveyor belt areas designated for experiment replication

In our replicated experiments, results demonstrated an appreciable enhancement in overall grasping performance. Our system achieved a success rate of 80% in comparison to the 73% recorded in Meile's experiments, and 92% vs. the 86% when excluding IK errors. This increase in success rate is largely attributable to the enhancements made to the system as detailed in the previous sections.

The failures that did occur were primarily due to two causes in our case: the gripper not closing upon moving to the always correctly calculated grasping position (it can be seen that, practically

100% of the time, the calculated grasping point is the optimal one) and persisting IK problems. Even with these difficulties, the improvements made to our model and system implementation helped to overcome some of the common pitfalls encountered in both studies (such as the IK problems). The higher overall success rate illustrates this fact and demonstrates the effectiveness of the changes made to our system.

In the following subsections, we will elaborate on the object-specific and location-specific success rates.

5.3.1. Success Rate per Object

When we examine the success rate stratified by object type, it becomes apparent that our model's performance exhibits a far lower dependency on the object in question when compared to Meile's experiments. This reduction in variability is likely attributable to the improvements we implemented to the overall system, enhancing its stability across a wide range of situations.

Considering all experiments, our model achieved an average success rate of approximately 80% compared to Meile's 73%. The success rates of the individual objects can be seen in the following figure:

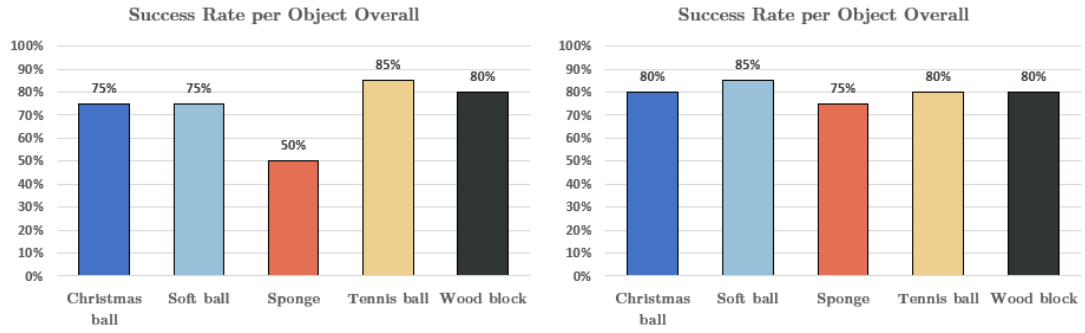

Figure 43. Success rate per object (overall) comparison: Meile's results (left) vs Ours (right)

If we further refine our analysis by excluding IK error attempts, the success rates for both projects show an expected increase. However, our model comes closest to perfection by consistently achieving around 92% success across all object types, compared to Meile's average success rate of 86%. It is pertinent to mention that in this refined subset, the total number of experiments reduced due to the exclusion of attempts with IK errors. This decreased the sample size from 100 to 85 in Meile's case and 87 in ours.

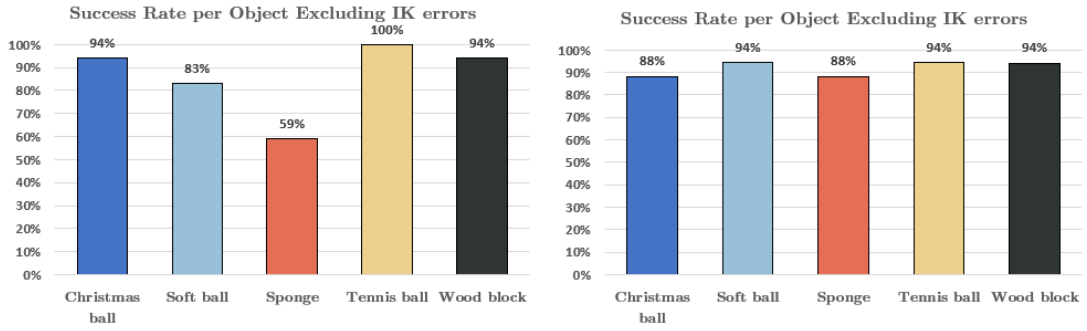

Figure 44. Success rate per object (excluding IK errors) comparison: Meile's results (left) vs Ours (right)

It is important to note that the consistent yet suboptimal success rate of around 92%, is largely due to instances in which the gripper did not close to grasp the object when it was supposed to, rather than issues related to the model's object recognition or decision-making processes.

In the upcoming subsection, we will delve into a location-specific analysis of success rates.

5.3.2. Success Rate per Area

When we analyse the success rates across different areas on the conveyor belt, it's clear that, both our experiments and Meile's ones, display a noticeable drop in performance in Area 1, and to a lesser extent in Area 2 (in our case). Nevertheless, our model proves its resilience by achieving a higher overall success rate of 80% compared to Meile's 73%.

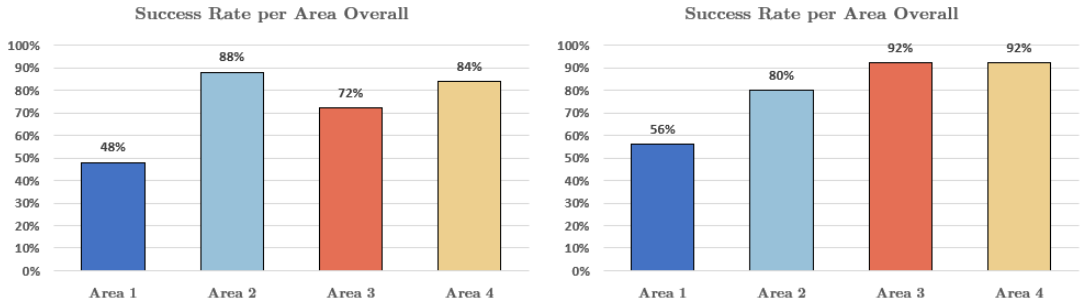

Figure 45. Success rate per area (overall) comparison: Meile's results (left) vs Ours (right)

Upon excluding attempts plagued by IK errors, our model's performance becomes remarkably consistent across all areas, averaging a success rate of approximately 92%. This finding underscore two main observations: Firstly, the known IK problems are particularly pronounced in Area 1 and, to a smaller degree, in Area 2. Secondly, the occasional issue of the gripper failing to close and grasp the object is consistently independent of the object's location on the conveyor belt. In contrast, Meile's performance, once adjusted for IK errors, reaches an average success rate of 86%.

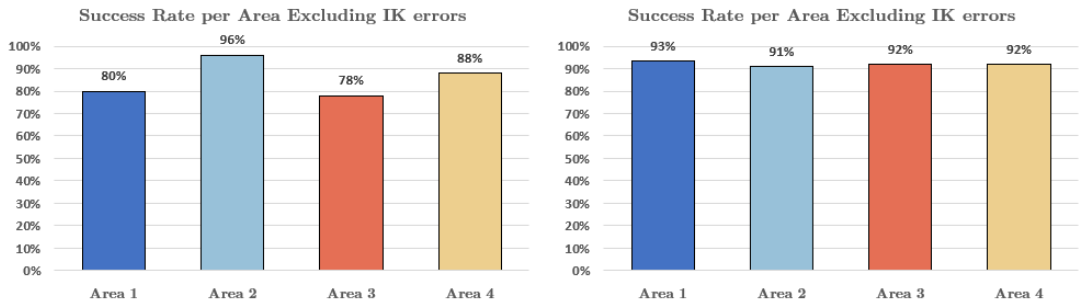

Figure 46. Success rate per area (excluding IK errors) comparison: Meile's results (left) vs Ours (right)

In essence, the success rate achieved by our model remains high and stable in all areas, demonstrating its ability to cope well with a wide variety of object positions, with a slightly improved success rate in the IK error areas as well. This result reflects the effectiveness of the improvements implemented in our system, including the strategic management of gripper rotation thanks to the effective use of decision trees, thus anticipating and mitigating potential failures.

6. Discussion

In this chapter, we will delve into a comprehensive discussion surrounding the potential application areas, the limitations encountered, and prospective future work based on the research and findings presented during this project. This will allow for a deeper understanding of the implications and possibilities arising from this study.

6.1. Potential application areas

The improvements made in our object grasping algorithm and system, as demonstrated in the context of a robot arm in a conveyor belt setup, have wide-ranging potential applications across several environments.

One of the most direct applications is within automated manufacturing and assembly lines. By enhancing the efficiency and success rate of the robotic arm's grasping capabilities, our system can help to increase overall productivity, reduce errors, and limit human intervention. The improvements made to the Calculation of the Optimal Grasping Point and the Grasp Stability Prediction using Machine Learning Techniques can lead to smoother operations in environments such as automotive assembly, electronics manufacturing and packaging facilities.

The advancements could also be of significant value in the realm of logistics and warehousing, where the automation of tasks like sorting, packaging, and moving items is rapidly increasing. The same principle could be applied to e-commerce distribution centres, where the demand for efficient and reliable automation solutions is constantly growing.

In addition, there are promising applications in the field of service robotics, for instance, in domestic, healthcare, or elderly care environments. Improving the precision and reliability of robotic grasping can help these service robots to perform daily tasks more effectively, contributing to enhanced quality of life for individuals requiring assistance.

6.2. Limitations

Despite the improvements made and the promising results achieved, there were limitations encountered during the course of this project.

The primary limitation was the persistence of IK errors, especially in certain areas of the conveyor belt. Although these errors were reduced to some extent through restricting the gripper rotation angle, they were not completely eliminated.

Another limitation was that the gripper occasionally failed to close upon reaching the correctly calculated grasping position. Despite the system's ability to nearly always calculate the optimal grasping point, there were instances where the gripper failed to act accordingly, leading to unsuccessful attempts.

6.3. Future Work

Building on the foundation established in this thesis, several paths for future work are evident. One clear direction is to further investigate and address the persistent issues relating to IK errors and the gripper's failure to close at times. This could involve studying the hardware elements involved, including the design of the gripper itself, or looking deeper into the software and control algorithms.

Another direction for future work would be to augment the system with further sensory inputs. This enhancement could incorporate depth perception and colour recognition capabilities, along with the deployment of more cameras, amongst other potential advancements. By taking advantage of these additional sensory inputs, the robotic arm's decision-making process and understanding of the environment could be enriched, enhancing the system's adaptability and efficiency. This multisensory approach has the potential to enhance the system's performance in a wide range of tasks, further strengthening its applicability in various fields.

Thanks to these improvements and upgrades, the usability and applicability of our robotic arm system can be further enhanced, expanding its potential in the fields of automation and robotics.

7. Conclusion

This thesis has presented a comprehensive study into improving the performance of a robotic arm in a conveyor belt setup through enhanced object grasping capabilities. The work performed builds on the foundation laid by previous research, focusing on the design of a new Calculation of the Optimal Grasping Point Calculation of the Optimal Grasping Point method and the utilization of machine learning techniques to predict grasp stability.

The Literature Review, as outlined in Chapter 2, provided an overview of the existing knowledge and methodologies in the field. It highlighted the pivotal role that machine learning and robotic automation play in modern manufacturing and logistics. However, it also underscored the need for continued research and innovation to overcome persisting challenges, particularly in optimizing robotic grasping techniques for enhanced efficiency and reliability.

Chapter 3's System Overview provided a detailed explanation of all different hardware and software components that we have been working on. Then, in Chapter 4, we analysed our newly devised method, the Calculation of the Optimal Grasping Point, which, together with the implementation of a Grasp Stability Prediction using Machine Learning Techniques, enabled the system to achieve better results in terms of success rate and stability.

Our Results, detailed in Chapter 5, demonstrated remarkable improvements achieved through our newly developed techniques. With an impressive success rate of 80% for simple-shaped objects (an astounding 92% if we discount IK errors), we have superseded the previous reference of 73% (86% without IK errors). Most notably, we have excelled in the complex field of intricately shaped objects as well, manifesting a success rate around 90% for bananas, for example, a striking elevation from the previous 18%. This significant leap forward is largely thanks to the innovative methodologies we have integrated into the system, which corroborates their efficacy.

Despite these promising results, we also encountered limitations in our study. Persistent IK errors and occasional gripper failures highlighted areas for future improvement. However, the consistency in the success rate across different objects indicates the effectiveness of the improvements made to the whole system, especially when calculating the best grasping point.

The implications of this research extend broadly to all sectors, especially automation and robotics, and promise greater efficiency and innovation. Our thesis contributes to the development of new techniques for enhanced object manipulation, which can be used by a variety of robots in several industries. Although challenges remain, our effort have laid a solid foundation for further exploration of robotics. With continuous improvement and the integration of more advanced sensory inputs and control algorithms, we foresee significant improvement of robotic capabilities in various sectors.

Bibliography

[1] H. Lasi, H. G. Kemper, P. Fettke, T. Feld and M. Hoffmann, "Industry 4.0," *BUSINESS & INFORMATION SYSTEMS ENGINEERING,* vol. 6, no. 4, pp. 239-242, 2014.

[2] Stanford University, "Fundamentals of Grasping," *Principles of Robot Autonomy,* 2023.

[3] United Nations, "The 17 goals," [Online]. Available: https://sdgs.un.org/goals. [Accessed July 2023].

[4] N. Meile, "Visual-Tactile Fusion for Intelligent Robotic Grasping," 2023.

[5] B. Siciliano and O. Khatib, Eds., Springer Handbook of Robotics, Springer Berlin, Heidelberg, 2008, pp. 671, 701-703.

[6] F. Almeida, A. Lopes and P. Abreu, "Force-Impedance Control: a new control strategy of robotic manipulators," IDMEC - Pólo FEUP, Faculdade de Engenharia da Universidade do Porto, 1999.

[7] A. Saxena, J. Driemeyer and A. Y. Ng, "Robotic Grasping of Novel Objects using Vision," The International Journal of Robotics Research, vol. 27, no. 2, p. 157–173, 2008.

[8] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez and T. Funkhouser, "Learning Synergies between Pushing and Grasping Learning Synergies between Pushing and Grasping," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018. [Online]. Available: https://vpg.cs.princeton.edu/.

[9] D. Park, Y. Seo and . S. Y. Chun, "Real-Time, Highly Accurate Robotic Grasp Detection using Fully Convolutional Neural Networks with High-Resolution Images," 2018. [Online]. Available: https://arxiv.org/pdf/1809.05828v1.pdf.

[10] H. Zhang, J. Tang, S. Sun and X. Lan, "Robotic Grasping from Classical to Modern: A Survey," 2022.

[11] C. K. Kruuse, "A ModelFree Predictive Control Framework for Trajectory Optimisation with a Collaborative Robot," 2022.

[12] F. Martín and M. Veloso, "Effective real-time visual object detection," *Progress in Artificial Intelligence,* vol. 1, no. 4, p. 259–265, 2012.

[13] X. Liu, B. Dai and H. He, "Real-time Object Segmentation for Visual Object Detection in Dynamic Scenes," Proceedings of the 2011 International Conference of Soft Computing and Pattern Recognition, Socpar, p. 423–428, 2011.

[14] H. Jiang and F. Liang, "Research on Image Fuzzy Edge Processing Based on Subpixel and Ramer-Douglas-Peucker Algorithm," *Ieee 4th International Conference on Civil Aviation Safety and Information Technology (iccasit),* p. 749–753, 2022.

[15] A. Menafoglio, "Principal Component Analysis," *Encyclopedia of Mathematical Geosciences,* pp. 1-4, 2020.

[16] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review,* vol. 39, pp. 261-283, 2013.

[17] Franka Emika, "FRANKA EMIKA ROBOT'S INSTRUCTION HANDBOOK," 2021.

[18] "ROS.org," [Online]. Available: http://wiki.ros.org/.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

A. Additional images on the calculation of the optimal grip point

Slope 1	Slope 2	Slope 2 - 1	Score	x1	y1	x2	y2
11,4591559	354,087917	342,628761	0,0965	1185	977	1196	977
11,4591559	354,087917	342,628761	0,0965	1186	977	1197	977
5,72957795	348,931297	343,201719	0,0933	1187	977	1198	977
5,72957795	348,931297	343,201719	0,0933	1188	977	1199	977
21,7723962	348,931297	327,158901	0,1825	1182	978	1200	978
16,6157761	348,931297	332,315521	0,1538	1183	978	1200	978
16,6157761	348,931297	332,315521	0,1538	1184	978	1200	978
26,3560586	338,045099	311,689041	0,2684	1180	979	1203	979
21,7723962	338,045099	316,272703	0,2429	1181	979	1203	979
26,3560586	333,461437	307,105378	0,2939	1178	980	1206	980
26,3560586	333,461437	307,105378	0,2939	1179	980	1206	980
30,9397209	328,877774	297,938053	0,3448	1176	981	1208	981
30,9397209	328,877774	297,938053	0,3448	1177	981	1208	981
34,9504255	321,429323	286,478898	0,4085	1175	982	1210	982
41,825919	321,429323	279,603404	0,4466	1173	983	1211	983
34,9504255	321,429323	286,478898	0,4085	1174	983	1211	983
41,825919	317,991576	276,165657	0,4657	1172	984	1212	984
45,2636658	317,991576	272,72791	0,4848	1171	985	1213	985
48,1284548	315,126787	266,998333	0,5167	1170	986	1214	986
48,1284548	312,261998	264,133544	0,5326	1169	987	1215	987
55,0039483	312,261998	257,25805	0,5708	1169	988	1216	988
55,0039483	312,261998	257,25805	0,5708	1168	989	1217	989
59,0146529	308,824252	249,809599	0,6122	1167	990	1218	990
59,0146529	304,813547	245,798894	0,6345	1167	991	1219	991
63,5983153	304,813547	241,215232	0,6599	1166	992	1219	992
63,5983153	300,802842	237,204527	0,6822	1166	993	1220	993
63,5983153	300,802842	237,204527	0,6822	1165	994	1221	994
68,1819776	296,792138	228,61016	0,7299	1165	995	1221	995
68,1819776	291,635518	223,45354	0,7586	1164	996	1221	996
73,3385978	291,635518	218,29692	0,7872	1164	997	1222	997
73,3385978	286,478898	213,1403	0,8159	1164	998	1222	998
78,4952179	281,322277	202,827059	0,8732	1163	999	1223	999
78,4952179	281,322277	202,827059	0,8732	1163	1000	1223	1000
84,2247959	281,322277	197,097482	0,905	1163	1001	1223	1001
84,2247959	275,592699	191,367904	0,9368	1163	1002	1223	1002
84,2247959	275,592699	191,367904	0,9368	1163	1003	1223	1003
89,9543738	269,863122	179,908748	0,9995	1163	1004	1223	1004
89,9543738	269,863122	179,908748	0,9995	1163	1005	1223	1005
89,9543738	269,863122	179,908748	0,9995	1163	1006	1223	1006
89,9543738	269,863122	179,908748	0,9995	1163	1007	1223	1007
89,9543738	269,863122	179,908748	0,9995	1163	1008	1223	1008
89,9543738	269,863122	179,908748	0,9995	1163	1009	1223	1009
89,9543738	269,863122	179,908748	0,9995	1163	1010	1223	1010
89,9543738	269,863122	179,908748	0,9995	1163	1011	1223	1011
89,9543738	269,863122	179,908748	0,9995	1163	1012	1223	1012
95,6839518	269,863122	174,17917	0,9677	1163	1013	1223	1013
95,6839518	269,863122	174,17917	0,9677	1163	1014	1223	1014
95,6839518	264,133544	168,449592	0,9358	1163	1015	1223	1015
101,41353	264,133544	162,720014	0,904	1163	1016	1223	1016
101,41353	264,133544	162,720014	0,904	1163	1017	1223	1017
106,57015	258,403966	151,833816	0,8435	1164	1018	1223	1018
106,57015	258,403966	151,833816	0,8435	1164	1019	1223	1019
111,72677	253,247345	141,520575	0,7862	1164	1020	1222	1020
111,72677	248,090725	136,363955	0,7576	1165	1021	1222	1021
116,310432	248,090725	131,780293	0,7321	1165	1022	1222	1022
111,72677	243,507063	131,780293	0,7321	1166	1023	1221	1023
116,310432	238,923401	122,612968	0,6812	1166	1024	1221	1024
120,894095	238,923401	118,029306	0,6557	1167	1025	1220	1025
120,894095	234,912696	114,018601	0,6334	1167	1026	1219	1026
124,904799	231,474949	106,57015	0,5921	1168	1027	1219	1027
124,904799	231,474949	106,57015	0,5921	1168	1028	1218	1028

Figure 47. Difference of slopes of each evaluated pair of points and resulting score for a ball

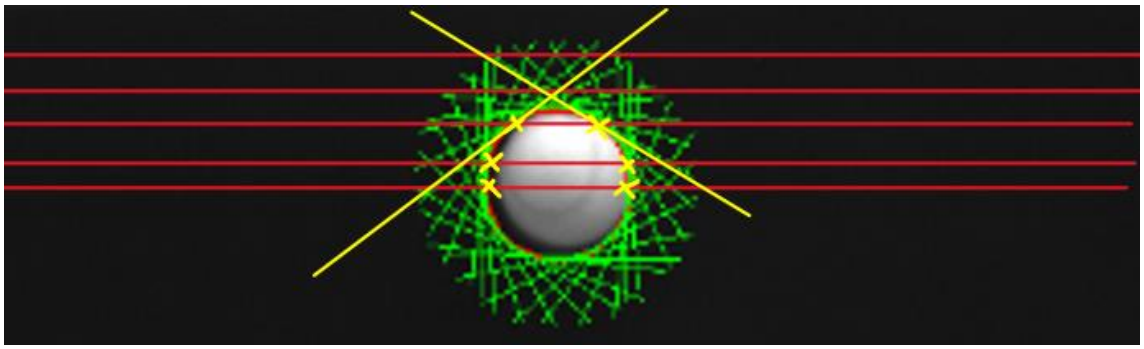

Figure 48. Ball used for the evaluation of the scoring function results

	x1	y1	x2	y2	Slope_score	dist_to_center_score	point_dist_score	score
2	1185	977	1196	977	0,0965	0,0268	0,8736	0,231
3	1182	978	1200	978	0,1825	0,0585	0,7931	0,2674
4	1180	979	1203	979	0,2684	0,0901	0,7356	0,3084
5	1178	980	1206	980	0,2939	0,1214	0,6782	0,319
6	1176	981	1208	981	0,3448	0,1522	0,6322	0,3445
7	1175	982	1210	982	0,4085	0,1833	0,5977	0,3788
8	1173	983	1211	983	0,4466	0,2138	0,5632	0,4001
9	1172	984	1212	984	0,4657	0,2446	0,5402	0,4143
10	1171	985	1213	985	0,4848	0,2754	0,5172	0,4285
11	1170	986	1214	986	0,5167	0,3062	0,4943	0,4491
12	1169	987	1215	987	0,5326	0,337	0,4713	0,4617
13	1169	988	1216	988	0,5708	0,3682	0,4598	0,4878
14	1168	989	1217	989	0,5708	0,399	0,4368	0,4925
15	1167	990	1218	990	0,6122	0,4298	0,4138	0,5178
16	1167	991	1219	991	0,6345	0,4606	0,4023	0,5359
17	1166	992	1219	992	0,6599	0,4914	0,3908	0,5555
18	1166	993	1220	993	0,6822	0,5222	0,3793	0,5736
19	1165	994	1221	994	0,6822	0,553	0,3563	0,5783
20	1165	995	1221	995	0,7299	0,5838	0,3563	0,6113
21	1164	996	1221	996	0,7586	0,6146	0,3448	0,6326
22	1164	997	1222	997	0,7872	0,6454	0,3333	0,6539
23	1164	998	1222	998	0,8159	0,6762	0,3333	0,6775
24	1163	999	1223	999	0,8732	0,707	0,3103	0,7108
25	1163	1000	1223	1000	0,8732	0,7378	0,3103	0,72
26	1163	1001	1223	1001	0,905	0,7686	0,3103	0,7451
27	1163	1002	1223	1002	0,9368	0,7994	0,3103	0,7703
28	1163	1003	1223	1003	0,9368	0,8302	0,3103	0,7795
29	1163	1004	1223	1004	0,9995	0,861	0,3103	0,8201
30	1163	1005	1223	1005	0,9995	0,8917	0,3103	0,8293
31	1163	1006	1223	1006	0,9995	0,9224	0,3103	0,8385
32	1163	1007	1223	1007	0,9995	0,953	0,3103	0,8477
33	1163	1008	1223	1008	0,9995	0,9827	0,3103	0,8566
34	X	1009	1223	1009	0,9995	0,983	0,3103	0,8567

Figure 49. Visualisation of the 3 scoring function values and the total weighted score for all evaluated pair of points of the ball - Image 1 of 2

1	x1	y1	x2	y2	Slope_score	dist_to_center_score	point_dist_score	score
35	1163	1010	1223	1010	0,9995	0,9533	0,3103	0,8478
36	1163	1011	1223	1011	0,9995	0,9228	0,3103	0,8386
37	1163	1012	1223	1012	0,9995	0,8921	0,3103	0,8294
38	1163	1013	1223	1013	0,9677	0,8613	0,3103	0,8043
39	1163	1014	1223	1014	0,9677	0,8305	0,3103	0,7951
40	1163	1015	1223	1015	0,9358	0,7998	0,3103	0,7699
41	1163	1016	1223	1016	0,904	0,769	0,3103	0,7448
42	1163	1017	1223	1017	0,904	0,7382	0,3103	0,7355
43	1164	1018	1223	1018	0,8435	0,7066	0,3218	0,6981
44	1164	1019	1223	1019	0,8435	0,6758	0,3218	0,6888
45	1164	1020	1222	1020	0,7862	0,6458	0,3333	0,6535
46	1165	1021	1222	1021	0,7576	0,6144	0,3448	0,6321
47	1165	1022	1222	1022	0,7321	0,5836	0,3448	0,6101
48	1166	1023	1221	1023	0,7321	0,5528	0,3678	0,6054
49	1166	1024	1221	1024	0,6812	0,5221	0,3678	0,5708
50	1167	1025	1220	1025	0,6557	0,4913	0,3908	0,5534
51	1167	1026	1219	1026	0,6334	0,4609	0,4023	0,5354
52	1168	1027	1219	1027	0,5921	0,4297	0,4138	0,5077
53	1168	1028	1218	1028	0,5921	0,3993	0,4253	0,5009
54	1169	1029	1217	1029	0,5507	0,3685	0,4483	0,4756
55	1170	1030	1216	1030	0,5507	0,3377	0,4713	0,4709
56	1171	1031	1215	1031	0,5157	0,3069	0,4943	0,4488
57	1172	1032	1214	1032	0,4806	0,2761	0,5172	0,4266
58	1173	1033	1213	1033	0,4456	0,2453	0,5402	0,4044
59	1174	1034	1212	1034	0,4297	0,2145	0,5632	0,3918
60	1175	1035	1211	1035	0,4106	0,1837	0,5862	0,3777
61	1177	1036	1209	1036	0,3438	0,1528	0,6322	0,3442
62	1179	1037	1207	1037	0,296	0,122	0,6782	0,3202
63	1181	1038	1204	1038	0,2674	0,0912	0,7356	0,3082
64	1183	1039	1201	1039	0,1846	0,0602	0,7931	0,269
65	1186	1040	1197	1040	0,0955	0,0289	0,8736	0,2311

Figure 50. Visualisation of the 3 scoring function values and the total weighted score for all evaluated pair of points of the ball - Image 2 of 2

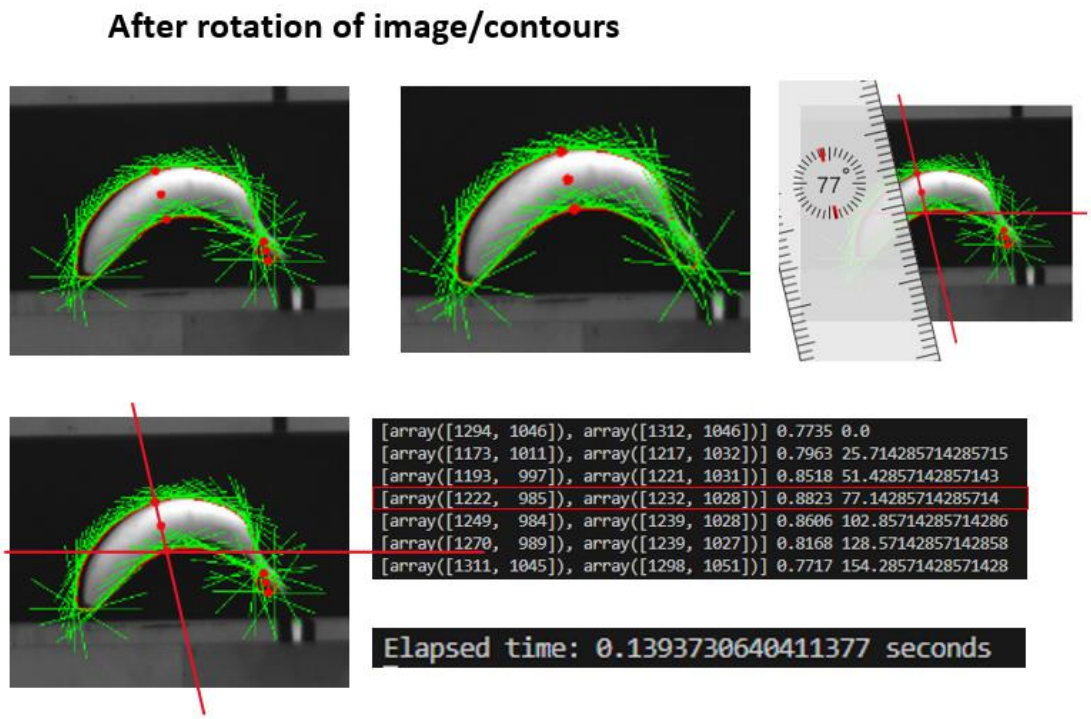

Figure 52. Evaluation of the total weighted score values for different rotation and its computational time required

```
(868, 1033, 0, 0, 2.79, 0), (870, 1037, 0, 0, 2.09, 0)] counter: [13, 8, 8, 8, 7, 14, 1, 1, 1]
Best point overall: None
[(868, 1033, 0, 0, 2.09, 0), (871, 1035, 0, 0, 2.09, 0), (870, 1036, 0, 0, 2.09, 0), (871, 1036, 0, 0, 2.09, 0), (869, 1034, 0, 0, 2.09, 0), (869, 1035, 0, 0, 2.44, 0), (870, 1035, 0, 0, 2.09, 0),
(868, 1033, 0, 0, 2.79, 0), (870, 1037, 0, 0, 2.09, 0)] counter: [14, 8, 8, 8, 7, 14, 1, 1, 1]
Best point overall: None
[(868, 1033, 0, 0, 2.09, 0), (871, 1035, 0, 0, 2.09, 0), (870, 1036, 0, 0, 2.09, 0), (871, 1036, 0, 0, 2.09, 0), (869, 1034, 0, 0, 2.09, 0), (869, 1035, 0, 0, 2.44, 0), (870, 1035, 0, 0, 2.09, 0),
(868, 1033, 0, 0, 2.79, 0), (870, 1037, 0, 0, 2.09, 0)] counter: [14, 9, 8, 8, 7, 14, 1, 1, 1]
Best point overall: None
[(868, 1033, 0, 0, 2.09, 0), (871, 1035, 0, 0, 2.09, 0), (870, 1036, 0, 0, 2.09, 0), (871, 1036, 0, 0, 2.09, 0), (869, 1034, 0, 0, 2.09, 0), (869, 1035, 0, 0, 2.44, 0), (870, 1035, 0, 0, 2.09, 0),
(868, 1033, 0, 0, 2.79, 0), (870, 1037, 0, 0, 2.09, 0)] counter: [14, 9, 8, 8, 7, 14, 2, 1, 1]
Best point overall: None
[(868, 1033, 0, 0, 2.09, 0), (871, 1035, 0, 0, 2.09, 0), (870, 1036, 0, 0, 2.09, 0), (871, 1036, 0, 0, 2.09, 0), (869, 1034, 0, 0, 2.09, 0), (869, 1035, 0, 0, 2.44, 0), (870, 1035, 0, 0, 2.09, 0),
(868, 1033, 0, 0, 2.79, 0), (870, 1037, 0, 0, 2.09, 0)] counter: [14, 9, 8, 8, 7, 14, 2, 1, 2]
Best point overall: None
[(868, 1033, 0, 0, 2.09, 0), (871, 1035, 0, 0, 2.09, 0), (870, 1036, 0, 0, 2.09, 0), (871, 1036, 0, 0, 2.09, 0), (869, 1034, 0, 0, 2.09, 0), (869, 1035, 0, 0, 2.44, 0), (870, 1035, 0, 0, 2.09, 0),
(868, 1033, 0, 0, 2.79, 0), (870, 1037, 0, 0, 2.09, 0)] counter: [14, 9, 8, 9, 7, 14, 2, 1, 2]
Best point overall: None
[(868, 1033, 0, 0, 2.09, 0), (871, 1035, 0, 0, 2.09, 0), (870, 1036, 0, 0, 2.09, 0), (871, 1036, 0, 0, 2.09, 0), (869, 1034, 0, 0, 2.09, 0), (869, 1035, 0, 0, 2.44, 0), (870, 1035, 0, 0, 2.09, 0),
(868, 1033, 0, 0, 2.79, 0), (870, 1037, 0, 0, 2.09, 0), (870, 1035, 0, 0, 2.79, 0)] counter: [14, 9, 8, 8, 9, 7, 14, 2, 1, 2, 1]
Best point overall: None
[(868, 1033, 0, 0, 2.09, 0), (871, 1035, 0, 0, 2.09, 0), (870, 1036, 0, 0, 2.09, 0), (871, 1036, 0, 0, 2.09, 0), (869, 1034, 0, 0, 2.09, 0), (869, 1035, 0, 0, 2.44, 0), (870, 1035, 0, 0, 2.09, 0),
(868, 1033, 0, 0, 2.79, 0), (870, 1037, 0, 0, 2.09, 0), (870, 1035, 0, 0, 2.79, 0)] counter: [14, 9, 8, 8, 10, 7, 14, 2, 1, 2, 1]
Best point overall: None
[] counter: []
Best point overall: (869, 1035, 0, 0, 2.44, 0)
[ WARN] [1686676681.923936127]: Going to x=0.48 y=-0.12 z=0.27 oz=5.58
[ INFO] [1686676688.891911877]: Task 2 finished.
[ INFO] [1686676693.443954584]: Task 3 finished.
[ INFO] [1686676695.899988731]: Task 5 finished.
[INFO] [1686676698.018598]: 1.62
[ INFO] [1686676698.031918020]: Task 6 finished.
[ INFO] [1686676700.835877067]: Task 7 finished.
[ INFO] [1686676707.215949245]: Task 8 finished.
```

Figure 51. Evaluation of the correct sending of the calculated best grasping point ("Best point overall") to the robot once the same best-calculated point reaches a count of 15